Correcting Multiple Deletions and Insertions in Racetrack Memory

Jin Sima and Jehoshua Bruck, *Fellow, IEEE*

***Abstract*—Racetrack memory is a tape-like structure where data is stored sequentially as a track of single-bit memory cells. The cells are accessed through read/write ports, called heads. When reading/writing the data, the heads stay fixed and the track is shifting. One of the main challenges in developing racetrack memory systems is the limited precision in controlling the track shifts, that in turn affects the reliability of reading and writing the data. A current proposal for combating deletions in racetrack memories is to use redundant heads per-track resulting in multiple copies (potentially erroneous) and recovering the data by solving a specialized version of a sequence reconstruction problem. Using this approach, k-deletion correcting codes of length n, with $d \geq 2$ heads per-track, with redundancy $\log\log n + 4$ were constructed. However, the known approach requires that $k \leq d$, namely, that the number of heads d is larger than or equal to the number of correctable deletions k. Here we address the question: What is the asymptotically optimal order of redundancy that can be achieved for a k-deletion code (k is a constant) if the number of heads is fixed at d (due to implementation constraints)? One of our key results is an answer to this question, namely, we construct codes that can correct k deletions, for any k beyond the known limit of d. The codes have asymptotically $8k\log\log n+o(\log\log n)$ redundancy for $d \leq k \leq 2d-1$. In addition, when $k \geq 2d$, our codes have asymptotically $2\lfloor k/d\rfloor \log n + o(\log n)$ redundancy, that we prove it is order-wise optimal, specifically, we prove that the redundancy required for correcting k deletions is at least $\lfloor k/2d\rfloor \log n + o(\log n)$. The encoding/decoding complexity of our codes is $O(n\log^{2k+1} n)$. Finally, we ask a general question: What is the order-wise optimal redundancy for codes correcting a combination of at most k deletions and insertions in a d-head racetrack memory? We prove that the redundancy used for a combination of k deletion and insertion errors is asymptotically the same as that needed in the case of k deletion errors.**

***Index Terms*—Racetrack memory, Deletion codes.**

I. Introduction

Racetrack memory is a promising non-volatile memory that possesses the advantages of ultra-high storage density and low latency (comparable to SRAM latency) [9], [13]. It has a tape-like structure where the data is stored sequentially as a track of single-bit memory cells. The cells are accessed through read/write ports, called heads. When reading/writing the data, the heads stay fixed and the track shifts. Fig. 1 illustrates multiple heads reading/writing on a single track.

One of the main challenges in developing racetrack memory systems is the limited precision in controlling the track shifts, that in turn affects the reliability of reading and writing the data [6], [18]. Specifically, the track may either not shift or shift more steps than expected. When the track does not shift, the same cell is read twice, causing a sticky insertion. When the track shifts more than a single step, cells are skipped, causing deletions in the reads [3].

It is natural to use deletion and sticky insertion correcting codes to deal with the shift errors. Also, it is known that a code correcting deletions is capable of correcting an equal number of combination of deletions and insertions [7]. However, designing redundancy and complexity efficient deletion correcting codes has been an open problem for decades, though there is a significant advance toward the solution recently. In fact, for k, a constant number of deletions, and n, the code length, no explicit k-deletion correcting codes with redundancy less than $O(\sqrt{n})$[1] were known even for $k = 2$ until [1] proposed a code with redundancy asymptotically $64k^2\log k\log n+o(\log n)$. Evidently, the redundancy of this code is orders of magnitude away from the optimal, known to be in the range $k\log n+o(\log n)$ to $2k\log n+o(\log n)$ [7]. After [1], the work of [5] and [10] independently proposed k-deletion codes with $O(k\log n)$ bits of redundancy, which are order-wise optimal. Following [10], [11] proposed a systematic deletion code with $4k\log n+o(\log n)$ bits of redundancy and is computationally efficient for constant k. The redundancy was later improved in [12] to $(4k-1)\log n+o(\log n)$. Despite the recent progress in deletion and insertion correcting codes, it is still tempting to explore constructions of deletion and insertion correcting codes that are specialized for racetrack memories and provide more efficient redundancy and lower complexity encoding/decoding algorithms.

There are two approaches for constructing deletion/insertion correcting codes for racetrack memories. The first is to assume reading multiple parallel tracks simultaneously with a single head per-track. Based on this assumption, the proposed codes in [15] can correct up to two deletions per head and the proposed codes in [2] can correct l bursts of deletions, each of length at most b. The codes in [2] are asymptotically (in the number of heads) rate-optimal. The second approach is to

Manuscript received July 17, 2022; revised March 09, 2023; accepted May 8, 2023. This research was partially supported by NSF grant CCF-1816965 and NSF grant CCF-1717884. This paper was presented in part at the 2019 IEEE International Syposium on Information Theory. [DOI: 10.1109/ISIT.2019.8849783] *(Corresponding author: Jin Sima.)*

J. Sima was with the Electrical Engineering Department, California Institute of Technology, Pasadena, CA, 91125 USA. He is now with the Department of Electrical and Computer Engineering, University of Illinois Urbana-Champaign, Urbana 61801, IL, USA (e-mail: jsima@illinois.edu).

J. Bruck is with the Electrical Engineering Department, California Institute of Technology, Pasadena, CA, 91125 USA (e-mail: bruck@caltech.edu)

Communicated by A. Wachter-Zeh, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2023.3279766.

Digital Object Identifier 10.1109/TIT.2023.3279766

[1]Throughout this paper, we say $f(n) = O(g(n))$ if there exist two constants C_1 and C_2 and a positive integer n_0 such that $C_1 \leq \frac{f(n)}{g(n)} \leq C_2$ for $n \geq n_0$. In addition, we say $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

leverage the fact that one can add redundant heads to the same track with small area cost [3], [4], [16], [18]. As shown in Fig. 1, a track is read by multiple heads, resulting in multiple copies (potentially erroneous) of the same sequence. The heads stay fixed and normally adjacent heads have equal distance [18]. Hence, the deletion indices (the indices where deletions occur) in different reads have fixed relative distances. This can be regarded as a sequence reconstruction problem, where a sequence $\mathbf{c}$ needs to be recovered from multiple copies, each obtained after k deletions in $\mathbf{c}$. We emphasize that the general sequence reconstruction problem [8] is different from the multi-head racetrack memory settings, since in multi-head racetrack memory, the set of deletion indices in one head is a shift of that in another head [3]. Demonstrating the advantage of multiple heads, the paper [4] proposed an efficient k-head k-deletion code of length n with redundancy $\log\log n+4$ where the distance between adjacent heads, denoted by t, is at least $2\log n+4$, and a k-head $(k-1)$-deletion code with $O(1)$ redundancy where the head distance t is at least $(k(k-1)/2+1)\log n+(k^3+5k+3)/3$. The number of heads, denoted by d[2], used in the codes in [4] is required to be at least k. For k-deletion correcting codes using $d\leq k-1$ heads, constructions were proposed in [3] that use a classical single head k-deletion code as building blocks, which have redundancy at least $O((k-d)\log n)$. The head distance t in [3] is at least $(k(k-1)/2+1)\log n+(k^3+5k+3)/3$. It is known that the number of heads affects the area overhead of the racetrack memory device [18], hence, it motivates the following **natural question**: What is the asymptotically optimal order of redundancy that can be achieved for a k-deletion code (k is a constant) if the number of heads is fixed at d (due to area limitations)?

One of our **key results** is an answer to this question, namely, we construct length n codes that correct k deletions, for any k beyond the known limit of d. Our code has $8k\log\log n+o(\log\log n)$ redundancy for the case when $d\leq k\leq 2d-1$. In addition, when $k\geq 2d$, the code has $2\lfloor k/d\rfloor\log n+o(\log n)$ redundancy. The head distance t in our codes is at least $O(poly(k)\log n)$. Our key result is summarized formally by the following theorem. Notice that the theorem implies that the redundancy of our codes is asymptotically larger than optimal by a factor of at most four when $t=n^{o(1)}$ and $k\geq 2d$.

Theorem 1. *For constant positive integers k and d, let the distance t between adjacent heads be*

$$t\geq\max\{(3k+\lceil\log n\rceil+2)[k(k-1)/2+1] \\ +(7k-k^3)/6,(4k+1)(5k+\lceil\log n\rceil+3)\}. \quad (1)$$

Then for $d\leq k\leq 2d-1$, there exists an explicit length $N=n+8k\log t+o(\log t)$ d-head k-deletion correcting code with redundancy $8k\log t+o(\log t)$. Specifically, the redundancy is $8k\log\log n+o(\log\log n)$ for $t=O(poly(k)\log n)$ satisfying (1). For $k\geq 2d$, there exists an explicit length $N=n+2\lfloor k/d\rfloor\max\{\log n,4k\log t\}+o(\log n)$ d-head k-deletion correcting code with redundancy $2\lfloor k/d\rfloor\max\{\log n,4k\log t\}+o(\log n)$. Specifically, the redundancy becomes $2\lfloor k/d\rfloor\log n+o(\log n)$ for $t=n^{o(1)}$ satisfying (1). The encoding and decoding functions can be computed in $O(nt^{2k+1})$ time. Moreover, for $k\geq 2d$ and $t=n^{o(1)}$, the amount of redundancy of a d-head k-deletion correcting code is lower bounded by $\lfloor k/2d\rfloor\log n+o(\log n)$.

[2]Throughout this paper, it is assumed that $d\geq 2$.

Since in addition to deletion errors, sticky insertion errors and substitution errors occur in racetrack memory, we are interested in codes that correct not only deletions, but a combination of deletion, sticky insertion, and substitution errors in a multiple head racetrack memory. However, in contrast to single head cases where a deletion code is also a deletion/insertion code, there is no such equivalence in multiple head racetrack memories. Correcting a combination of at most k deletions and sticky insertions in total turns out to be more difficult than correcting k deletion errors. It is not known whether the k-deletion code with $\log\log n+O(1)$ redundancy and the $(k-1)$-deletion code with $O(1)$ redundancy in [3] apply to a combination of deletion and sticky insertion errors in a k-head racetrack memory.

Our second result, which is the main result in this paper, provides an answer for such scenarios. We consider a more general problem of correcting a combination of deletions and insertions in a d-head racetrack memory, rather than deletions and sticky insertions, and show that the redundancy result for deletion cases extends to cases with a combination of deletions and insertions. Note that this covers the cases with deletion, insertion, and substitution errors, since a substitution is a deletion followed by an insertion.

Theorem 2. *For constant positive integers k and d, let the distance t between adjacent heads be*

$$t>(\frac{k^2}{4}+3k)(6k+\lceil\log n\rceil+3)+8k+\lceil\log n\rceil+3. \quad (2)$$

Then for $k<d$, there exists an explicit length $N=n+k+1+O(1)$ code correcting a combination of at most k insertions and deletions in a d-head racetrack memory with redundancy $k+1+O(1)$. The encoding and decoding complexity is $poly(n)$. For $d\leq k\leq 2d-1$, there exists a length $N=n+4k\log t+o(\log t)$ code correcting a combination of at most k insertions and deletions in a d-head racetrack memory with redundancy $8k\log t+o(\log t)$. Specifically, the redundancy is $8k\log\log n+o(\log\log n)$ for $t=O(poly(k)\log n)$ satisfying (2). Finally, when $d\geq 2d$, there exists a length $N=n+2\lfloor k/d\rfloor\max\{\log n,4k\log t\}+o(\log n)$ code that corrects a combination of at most k insertions and deletions in a d-head racetrack memory with redundancy $2\lfloor k/d\rfloor\max\{\log n,4k\log t\}+o(\log n)$. Specifically, the redundancy is $2\lfloor k/d\rfloor\log n+o(\log n)$ for $t=n^{o(1)}$ satisfying (2). The encoding and decoding functions can be computed in $O(nt^{2k+1})$ time.

Remark 1. *Theorem 2 improves the head distance lower bound in Theorem 1 when $k\geq 15$ and n is sufficiently large.*

Remark 2. *As mentioned above, the shift errors in racetrack memory include deletions and sticky insertions. The deletion/insertion codes in Theorem 2 are more general. However, there is asymptotically no additional redundancy cost in our*

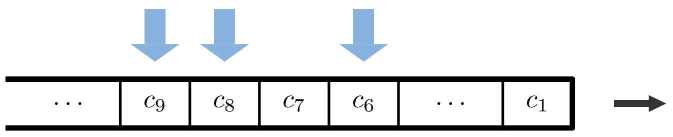

Fig. 1. Racetrack memory with multiple heads.

code constructions by considering the general deletions and insertions, since the redundancy of our deletion correcting only codes in Theorem 1 is asymptotically the same as that of the codes correcting deletions and insertions in Theorem 2. Moreover, the capability of correcting a combination of deletions and insertions makes it possible to correct a combination of deletions, insertions, and substitutions, a problem also studied in [3].

We note that the head distance t in our construction needs to be $O(poly(k)\log n)$ or $n^{o(1)}$ to get the desired redundancy results, while in [3], [4] the head distance t is only required to be larger than $O(poly(k)\log n)$ and can be $n^{O(1)}$. In racetrack memory systems [16], [18], the head distance between adjacent redundant heads can be small, and choosing larger head distance might increase the latency for decoding.

Some of the key ideas in our encoding/decoding include: (1) An efficient algorithm (described in Lemma 5) to map any binary sequence to a sequence such that its consecutive subsequences of given length are period free, which is used for encoding. (2) An algorithm (described in Section IV) to align (synchronize) the reads and identify the error indices under deletions, which is used for decoding. (3) An algorithm (described in Section VI) to align the reads and correct errors under deletions and insertions, used for decoding.

Organization: In Section II, we present the problem settings and some basic lemmas needed in our proof. Section III presents the proof of the main result for the case $d \leq k \leq 2d-1$. Section IV describes in detail how to synchronize the reads. The case $k \geq 2d$ is addressed in Section V. Section VI shows how to correct deletion and insertion errors and proves Theorem 2. Section VII concludes the paper.

II. Preliminaries

A. Problem Settings

We now describe the problem settings and the notations needed. For any two integers $i \leq j$, let $[i,j] = \{i, i+1, \ldots, j-1, j\}$ be an integer interval that contains all integers between i and j. Let $[i,j] = \emptyset$ for $i > j$. For a length n sequence $\mathbf{c} = (c_1, \ldots, c_n)$, an index set $\mathcal{I} \subseteq [1,n]$, let

$$\mathbf{c}_{\mathcal{I}} = (c_i : i \in \mathcal{I})$$

be a subsequence of $\mathbf{c}$, obtained by choosing the bits with indices in the set $\mathcal{I}$. Denote by $\mathcal{I}^c = [1,n]\backslash\mathcal{I}$ the complement of $\mathcal{I}$.

In the channel model of a d-head racetrack memory, the input is a binary sequence $\mathbf{c} \in \{0,1\}^n$. The channel output consists of d subsequences of $\mathbf{c}$ of length $n-k$, obtained by the d heads after k deletions in the channel input $\mathbf{c}$, respectively. Each subsequence is called a *read*. Let $\boldsymbol{\delta}_i = \{\delta_{i,1}, \ldots, \delta_{i,k}\} \subseteq [1,n]$ be the deletion indices in the ith head such that $\delta_{i,1} < \ldots < \delta_{i,k}$. Then, the read from the ith head is given by $\mathbf{c}_{\boldsymbol{\delta}_i^c}$, $i \in [1,d]$, i.e., bits c_ℓ, $\ell \in \boldsymbol{\delta}_i$ are deleted.

Note that in a d-head racetrack memory, the heads are placed in fixed locations, and the deletions are caused by "over-shifts" of the track. Hence when a deletion occurs at the jth bit in the read of the ith head, a deletion also occurs at the $(j+t)$th bit in the read of the $(i+1)$th head, $i \in [d-1]$, where t is the distance between adjacent heads. Then, the deletion index sets $\{\boldsymbol{\delta}_i\}_{i=1}^d$ satisfy

$$\boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t,$$

for a positive integer t, where for an integer set $\mathcal{S}$ and an integer t, $\mathcal{S} + t = \{x + t : x \in \mathcal{S}\}$.

To formally define a code for the d-head racetrack memory, we represent the d reads from the d heads by a $d \times (n-k)$ binary matrix, called the *read matrix*. The ith row of the read matrix is the read from the ith head. Let $\boldsymbol{D}(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d) \in \{0,1\}^{d\times(n-k)}$ be the read matrix of a d-head racetrack memory, where the input is $\mathbf{c} \in \{0,1\}^n$ and the deletion indices in the ith head are given by $\boldsymbol{\delta}_i$, $i \in [1,d]$. By this definition, the ith row of $\boldsymbol{D}(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d)$ is $\mathbf{c}_{\boldsymbol{\delta}_i^c}$.

Example 1. *Consider a 3 head racetrack memory with adjacent head distance $t = 2$. Let the deletion index set $\boldsymbol{\delta}_1 = \{2,5,6\}$. Then, we have that $\boldsymbol{\delta}_2 = \{4,7,8\}$ and $\boldsymbol{\delta}_3 = \{6,9,10\}$. Let $\mathbf{c} = (1,1,0,1,0,0,0,1,0,1)$ be a sequence of length 10. Then, the read matrix is given by*

$$\boldsymbol{D}(\mathbf{c}, \boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \boldsymbol{\delta}_3) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Remark 3. *We note that based on the model in this paper, no errors occur in the first $\delta_{1,1} + (i-1)t - 1$ bits of the ith head. Similarly, no errors occur in the last $n - \delta_{d,k} + (d-i)t$ bits. Though this assumption might simplify the cases in racetrack memories, where errors might occur in the first $\delta_{1,1} + (i-1)t - 1$ bits and the last $n - \delta_{d,k} + (d-i)t$ bits, we focus on this simplified assumption to study how multiple fixed heads help in correcting deletion/insertions. In addition, the first and the last bits in each head can be protected by adding extra redundancy.*

The deletion ball $\mathcal{D}_k(\mathbf{c}, t)$ of a sequence $\mathbf{c} \in \{0,1\}^n$ is the set of all possible read matrices in a d-head racetrack memory with input $\mathbf{c}$ and head distance t, i.e.,

$$\mathcal{D}_k(\mathbf{c}, t) = \{\boldsymbol{D}(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d) : \boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t, \boldsymbol{\delta}_i \subseteq [1,n], |\boldsymbol{\delta}_i| = k, i \in [1, d-1]\}.$$

A d-head k-deletion code $\mathcal{C}$ is the set of all sequences such that the deletion balls of any two do not intersect, i.e., for any $\mathbf{c}, \mathbf{c}' \in \mathcal{C}$, $\mathcal{D}_k(\mathbf{c}, t) \cap \mathcal{D}_k(\mathbf{c}', t) = \emptyset$.

The following notations will be used throughout the paper. For a matrix $\boldsymbol{A}$ and two index sets $\mathcal{I}_1 \subseteq [1,d]$ and $\mathcal{I}_2 \subseteq [1, n-k]$, let $\boldsymbol{A}_{\mathcal{I}_1,\mathcal{I}_2}$ denote the submatrix of $\boldsymbol{A}$ obtained by selecting the rows $i \in \mathcal{I}_1$ and the columns $j \in \mathcal{I}_2$. For any two integer sets $\mathcal{S}_1$ and $\mathcal{S}_2$, the set $\mathcal{S}_1 \backslash \mathcal{S}_2 = \{x : x \in \mathcal{S}_1, x \notin \mathcal{S}_2\}$ denotes the difference between sets $\mathcal{S}_1$ and $\mathcal{S}_2$.

A sequence $\mathbf{c} \in \{0,1\}^n$ is said to have period ℓ if $c_i = c_{i+\ell}$ for $i \in [1, n-\ell]$. We use $L(\mathbf{c}, \ell)$ to denote the length of the longest subsequence of consecutive bits in $\mathbf{c}$ that has period ℓ. Furthermore, define

$$L(\mathbf{c}, \leq k) \triangleq \max_{\ell \leq k} L(\mathbf{c}, \ell).$$

Example 2. *Let the sequence* $\mathbf{c}$ *be* $\mathbf{c} = (1,1,0,1,1,0,1,0,0)$. *Then we have that* $L(\mathbf{c},1) = 2$*, since* $\mathbf{c} = (\mathbf{1},\mathbf{1},0,\mathbf{1},\mathbf{1},0,1,\mathbf{0},\mathbf{0})$*, that* $L(\mathbf{c},2) = 4$*, since* $\mathbf{c} = (1,1,0,1,\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{0},0)$*, and that* $L(\mathbf{c},3) = 7$*, since* $\mathbf{c} = (\mathbf{1},\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{1},\mathbf{0},\mathbf{1},0,0)$*. Thus, we have* $L(\mathbf{c}, \leq 3) = 7$.

B. Racetrack Memory with Insertion and Deletion errors

We now describe the notations and problem settings for d-head racetrack memories with a combination of insertion and deletion errors, which is similar to d-head racetrack memories with deletion errors only. In addition to the deletion errors described by deletion index sets $\{\boldsymbol{\delta}_i\}_{i=1}^d$ satisfying

$$\boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t,$$

$i \in [1, d-1]$, and $|\boldsymbol{\delta}_i| = r$, $i \in [1,d]$, we consider insertion errors described by insertion index sets $\{\boldsymbol{\gamma}_i\}_{i=1}^d$ satisfying

$$\boldsymbol{\gamma}_{i+1} = \boldsymbol{\gamma}_i + t,$$

$i \in [1, d-1]$, where $\boldsymbol{\gamma}_i = \{\gamma_{i,1}, \ldots, \gamma_{i,s}\}$ for $i \in [1,d]$, and the inserted bits $\mathbf{b}_i = (b_{i,1}, b_{i,2}, \ldots, b_{i,s})$, $i \in [1,d]$. It is assumed that $\gamma_{i,j} \in [0,n]$ for $i \in [1,d]$ and $j \in [1,s]$. As a result of the insertion errors, the bit $b_{i,j}$ is inserted after the $\gamma_{i,j}$th bit of $\mathbf{c}$ in the ith head, for $i \in [1,d]$ and $j \in [1,s]$. When $\gamma_{i,j} = 0$, the insertion occurs before c_1 in the ith head. We note that $\mathbf{b}_i$ can be different for different i's.

We call a deletion error or an insertion error an *edit error*, or *error* in Section VI. For edit errors, define the read matrix $\boldsymbol{E}(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d, \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_d, \mathbf{b}_1, \ldots, \mathbf{b}_d) \in \{0,1\}^{d \times (n+s-r)}$, for $i \in [1,d]$, as follows. The ith row of $\boldsymbol{E}(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d, \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_d, \mathbf{b}_1, \ldots, \mathbf{b}_d) \in \{0,1\}^{d \times (n+s-r)}$ is obtained by deleting the bits $c_{\ell:\ell \in \boldsymbol{\delta}_i}$ and insert $b_{i,j}$ after $c_{\gamma_{i,j}}$, for $i \in [1,d]$ and $j \in [1,s]$. In this paper, we consider k edit errors. Hence, $r + s \leq k$.

Example 3. *(Follow-up of Example 1). Consider a* 3 *head racetrack memory with adjacent head distance* $t = 2$*. Let the deletion index set* $\boldsymbol{\delta}_1 = \{2,5,6\}$*. Then, we have that* $\boldsymbol{\delta}_2 = \{4,7,8\}$ *and* $\boldsymbol{\delta}_3 = \{6,9,10\}$*. In addition, the insertion index set is given by* $\boldsymbol{\gamma}_1 = \{0,2\}$*. Then, we have* $\boldsymbol{\gamma}_2 = \{2,4\}$*, and* $\boldsymbol{\gamma}_3 = \{4,6\}$*. Let* $\mathbf{b}_1 = (1,1)$*,* $\mathbf{b}_2 = (1,0)$*,* $\mathbf{b}_3 = (0,1)$*. Let* $\mathbf{c} = (1,1,0,1,0,0,0,1,0,1)$ *be a sequence of length* 10*. Then, the read matrix is given by*

$$\begin{aligned}&\boldsymbol{E}(\mathbf{c}, \boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \boldsymbol{\delta}_3, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\gamma}_3, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)\\ &= \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.\end{aligned}$$

Define an edit ball $\mathcal{E}_k(\mathbf{c}, t)$ of a sequence $\mathbf{c} \in \{0,1\}^n$ as the set of all possible read matrices in an d-head racetrack memory with input $\mathbf{c}$ and head distance t, i.e.,

$$\begin{aligned}\mathcal{E}_k(\mathbf{c}, t) = \{&\boldsymbol{E}(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d, \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_d, \mathbf{b}_1, \ldots, \mathbf{b}_d) :\\ &\boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t, \boldsymbol{\gamma}_{i+1} = \boldsymbol{\gamma}_i + t, \text{ for } i \in [1,d],\\ &\text{and } \boldsymbol{\delta}_i \subseteq [1,n], |\boldsymbol{\delta}_i| = r, \boldsymbol{\gamma}_i \subseteq [0,n], |\boldsymbol{\gamma}_i| = s,\\ &\mathbf{b}_i \in \{0,1\}^s \text{ for } i \in [1,d], r+s \leq k, \}.\end{aligned}$$

A d-head k edit correction code $\mathcal{C}$ is the set of all sequences such that the edit balls of any two do not intersect, i.e., for any $\mathbf{c}, \mathbf{c}' \in \mathcal{C}$, $\mathcal{E}_k(\mathbf{c}, t) \cap \mathcal{E}_k(\mathbf{c}', t) = \emptyset$.

C. Lemmas

In this section we present lemmas that will be used throughout the paper. Some of them are existing results. The following lemma describes a systematic Reed-Solomon code that can correct a constant number of erasures and can be efficiently computed (See for example [17]).

Lemma 1. *Let* q *be a power of* 2 *and let* k *and* n *be positive integers that satisfy* $n + k \leq q - 1$*. Then, there exists a map* $RS_k : \mathbb{F}_q^n \to \mathbb{F}_q^k$*, computable in* $poly(n)$ *time, such that* $\{(\mathbf{c}, RS_k(\mathbf{c})) : \mathbf{c} \in \mathbb{F}_q^n\}$ *is a* k *erasure correcting code.*

When q is the smallest power of 2 satisfying $n + k \leq q - 1$, the Reed-Solomon code requires redundancy $k \log q = k \log n + o(\log n)$ for correcting k erasures. Correcting a burst of two erasures requires less redundancy when the alphabet size of the code has order $o(\log n)$. The following code for correcting consecutive two erasures will be used for the case when the number of deletions k is less than $2d$.

Lemma 2. *Let* q *be a power of* 2 *and let* n *be a positive integer. There exists a map* $ER : \mathbb{F}_q^n \to \mathbb{F}_q^2$*, computable in* $O(n)$ *time, such that the code* $\{(\mathbf{c}, ER(\mathbf{c}) : \mathbf{c} \in \mathbb{F}_q^n\}$ *is capable of correcting two consecutive erasures.*

Proof. For a sequence $\mathbf{c} = (c_1, \ldots, c_n)$ Let the code ER be given by

$$ER(\mathbf{c}) = (\sum_{i=0}^{\lfloor (n-1)/2 \rfloor} c_{2i+1}, \sum_{i=0}^{\lfloor n/2 \rfloor} c_{2i}),$$

which are the sums of symbols with odd and even indices respectively over field $\mathbb{F}_q$. Note that two consecutive erasures are reduced to two single erasures, with one in the even symbols and one in the odd symbols, which can be recovered with the help of $ER(\mathbf{c})$. Hence, $(\mathbf{c}, ER(\mathbf{c}))$ can be recovered from two consecutive erasures. □

Our construction uses an explicit deletion code construction for a single read $d = 1$ as building blocks, which was presented in [11].

Lemma 3. *Let* k *be a fixed positive integer. For any positive integers* m *and* n*, there exists an explicit mapping*

$$H : \{0,1\}^m \to \{0,1\}^{\lceil 4k \log m + o(\log m) \rceil},$$

computable in $O(m^{2k+1})$ time, such that any sequence $\mathbf{c} \in \{0,1\}^m$ can be recovered from its length $m-k$ subsequence with the help of $H(\mathbf{c})$.

We also use the following fact, proved in [7], which implies that a deletion correcting code can be used to correct a combination of deletions and insertions.

Lemma 4. *A k-deletion correcting code is capable of correcting a combination of r deletions and s insertions, where $r+s \leq k$.*

Remark 4. *Note that the lemma does not hold in general in a multiple head racetrack memory considered in this paper.*

In addition, in order to synchronize the sequence $\mathbf{c}$ in the presence of deletions, we need to transform $\mathbf{c}$ to a sequence that has a limited length constraint on its periodic subsequences. Such constraint was considered in [3], where it was proved that the redundancy of the code $\{\mathbf{c} : L(\mathbf{c}, \leq k) \leq \lceil \log n \rceil + k + 1\}$ is at most 1 bit. In the following lemma we present a method to transform any sequence to one that satisfies the length constraint. The redundancy of our construction is $k+1$ bits. However, it is small compared to the redundancy of the d-head k-deletion code.

Lemma 5. *For any positive integers k and n, there exists an injective function $F : \{0,1\}^n \rightarrow \{0,1\}^{n+k+1}$, computable in $O_k(n^3 \log n)$ time, such that for any sequence $\mathbf{c} \in \{0,1\}^n$, we have that $L(F(\mathbf{c}), \leq k) \leq 3k + 2 + \lceil \log n \rceil$.*

Proof. Let $\mathbf{1}^x$ and $\mathbf{0}^y$ denote consecutive x 1's and consecutive y 0's, respectively. The encoding procedure for computing $F(\mathbf{c})$ is as follows.

1) **Initialization:** Let $F(\mathbf{c}) = \mathbf{c}$. Append $(\mathbf{1}^k, 0)$ to the end of the sequence $F(\mathbf{c})$. Let $i = 1$ and $n' = n$. Go to Step 1.
2) **Step 1:** If $i \leq n' - 2k - \lceil \log n \rceil - 1$ and $F(\mathbf{c})_{[i,i+2k+\lceil \log n \rceil+1]}$ has period $p \leq k$, let p_{min} be the smallest period of $F(\mathbf{c})_{[i,i+2k+\lceil \log n \rceil+1]}$. Delete $F(\mathbf{c})_{[i,i+2k+\lceil \log n \rceil+1]}$ from $F(\mathbf{c})$ and append $(\mathbf{1}^{k-p_{min}}, 0, F(\mathbf{c})_{[i,i+p_{min}-1]}, i, \mathbf{0}^{k+1})$ to the end of $F(\mathbf{c})$, i.e., set $F(\mathbf{c})_{[i,n-k-\lceil \log n \rceil-1]} = F(\mathbf{c})_{[i+2k+\lceil \log n \rceil+2,n+k+1]}$ and $F(\mathbf{c})_{[n-k-\lceil \log n \rceil,n+k+1]} = (\mathbf{1}^{k-p_{min}}, 0, F(\mathbf{c})_{[i,i+p_{min}-1]}, i, \mathbf{0}^{k+1})$. Let $n' = n' - 2k - \lceil \log n \rceil - 2$ and $i = 1$. Repeat. Else go to Step 2.
3) **Step 2:** If $i \leq n' - 2k - \lceil \log n \rceil - 1$, let $i = i+1$ and go to Step 1. Else output $F(\mathbf{c})$.

It can be verified that the length of the sequence $F(\mathbf{c})$ remains to be $n+k+1$ during the procedure. The number n' in the procedure denotes the number such that $F(\mathbf{c})_{[n'+1,n+k+1]}$ are appended bits and $F(\mathbf{c})_{[1,n']}$ are the remaining bits in $\mathbf{c}$ after deletions. Since either i increases to n' or n' decreases in Step 1. The algorithm terminates within $O(n^2)$ times of Step 1 and Step 2. Since it takes $O(k(3k+2+\log n)n)$ time to check the periodicity in Step 1. The total complexity is $O_k(n^3 \log n)$.

We now prove that $L(F(\mathbf{c}), \leq k) \leq 3k + 2 + \lceil \log n \rceil$. Let n' be the number computed in the encoding procedure. According to the encoding procedure, we have that $L(F(\mathbf{c})_{[j,j+2k+1+\lceil \log n \rceil]}, \leq k) \leq 2k + 1 + \lceil \log n \rceil$ for $j \leq n' - 2k - \lceil \log n \rceil - 1$, since any subsequence $F(\mathbf{c})_{[j,j+2k+1+\lceil \log n \rceil]}$ with period not greater than k is deleted. Therefore $L(F(\mathbf{c})_{[j,j+3k+1+\lceil \log n \rceil]}, \leq k) \leq 3k + 1 + \lceil \log n \rceil$ for $j \leq n' - 2k - \lceil \log n \rceil - 1$. For $n' - 2k - \lceil \log n \rceil \leq j \leq n'$, the sequence $F(\mathbf{c})_{[j,j+2k+1+\lceil \log n \rceil]}$ contains $F(\mathbf{c})_{[n'+1,n'+k+1]} = (\mathbf{1}^k, 0)$, which does not have period not greater than k. Hence we have that $L(F(\mathbf{c})_{[j,j+3k+1+\lceil \log n \rceil]}, \leq k) \leq 3k + 1 + \lceil \log n \rceil$. For $j > n'$, the sequence $F(\mathbf{c})_{[j,j+3k+1+\lceil \log n \rceil]}$ contains $\mathbf{0}^{k+1}$ as $k+1$ consecutive bits. Hence, if $L(F(\mathbf{c})_{[j,j+3k+1+\lceil \log n \rceil]}, \leq k) = 3k + 2 + \lceil \log n \rceil$, we have that $F(\mathbf{c})_{[j,j+3k+1+\lceil \log n \rceil]} = \mathbf{0}^{3k+2+\lceil \log n \rceil}$. However, this is impossible since $F(\mathbf{c})_{[j,j+3k+1+\lceil \log n \rceil]}$ contains either the index i to the left of $\mathbf{0}^{k+1}$ or the bits $(\mathbf{1}^{k-p_{min}}, 0, F(\mathbf{c})_{[i,i+p_{min}-1]})$ to the right of $\mathbf{0}^{k+1}$, both of which can not be all zero. Therefore, we conclude that $L(\mathbf{c}, \leq k) \leq 3k + 2 + \lceil \log n \rceil$. Given $F(\mathbf{c})$, the decoding procedure for computing $\mathbf{c}$ is as follows.

1) **Initialization:** Let $\mathbf{c} = F(\mathbf{c})$ and go to Step 1.
2) **Step 1:** If $\mathbf{c}_{[n+1,n+k+1]} \neq (\mathbf{1}^k, 0)$, let j be the length of the first 1 run in $\mathbf{c}_{[n-k-\lceil \log n \rceil,n+k+1]}$ and let p be the decimal representation of $\mathbf{c}_{n-\lceil \log n \rceil+1,n}$. Let $\mathbf{a}$ be a sequence of length $2k + \lceil \log n \rceil + 2$ and period $k - j$. The first $k - j$ bits of $\mathbf{a}$ is given by $\mathbf{c}_{[n-k-\lceil \log n \rceil+j+1,n-\lceil \log n \rceil]}$. Delete $\mathbf{c}_{[n-k-\lceil \log n \rceil,n+k+1]}$ from $\mathbf{c}$ and insert $\mathbf{a}$ at index p of $\mathbf{c}$, i.e., let $\mathbf{c}_{[p+2k+\lceil \log n \rceil+2,n+k+1]} = \mathbf{c}_{[p,n-k-\lceil \log n \rceil-1]}$ and $\mathbf{c}_{[p,p+2k+\lceil \log n \rceil+1]} = \mathbf{a}$. Repeat. Else output $\mathbf{c}$

Note that the encoding procedure consists of a series of deleting and appending operations. The decoding procedure consists of a series of deletion and inserting operations. Let $F_i(\mathbf{c})$, $i \in [0, R]$ be the sequence $F(\mathbf{c})$ obtained after the ith deleting and appending operation in the encoding procedure, where R is the number of deleting and appending operations in total in the encoding procedure. We have that $F_0(\mathbf{c}) = \mathbf{c}$ and $F_R(\mathbf{c})$ is the final output $F(\mathbf{c})$. It can be seen that the decoding procedure obtains $F_{R-i}(\mathbf{c})$, $i \in [0, R]$ after the ith deleting and inserting operation. Hence the function $F(\mathbf{c})$ is injective. □

Finally, we restate one of the main results in [3] that will be used in our construction. The result guarantees a procedure to correct $d-1$ deletions in a d-head racetrack memory, given that the distance between consecutive heads are large enough.

Lemma 6. *Let k and $d \leq k$ be two positive integers and let $\mathcal{C}$ be a $(k-d+1)$-deletion code. Then for any positive integer ℓ, $\mathcal{C} \cap \{\mathbf{c} : L(\mathbf{c}, \leq k) \leq \ell\}$ is a d-head k-deletion correcting code, given that the distance between consecutive heads $t \geq \ell[k(k-1)/2 + 1] + (7k - k^3)/6$.*

III. Correcting up to $2d-1$ deletions with d heads

In this section we construct a d-head k-deletion code for cases when $d \leq k \leq 2d - 1$. To this end, we first present a lemma that is crucial in our code construction. The lemma states that the range of deletion indices can be narrowed down to a list of short intervals. Moreover, the number of deletions

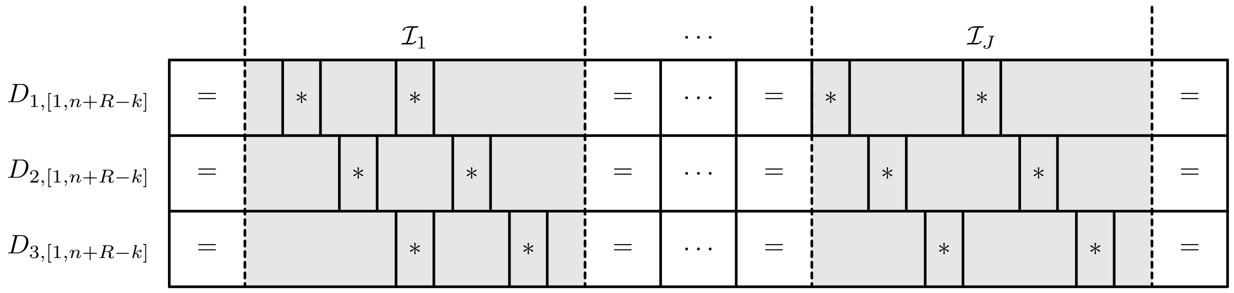

Fig. 2. An illustration of Lemma 7. The $*$ entries denote deletion in the heads. The read $D_{i,[1,n+R]}$ in each head is obtained after deleting the $*$ entries from $\mathbf{c}$.

within each interval can be determined. The proof of the lemma will be given in Section IV. Before presenting the lemma, we give the following definition, which describes a property of the intervals we look for.

Definition 1. *Let $\boldsymbol{\delta}_i = \{\delta_{i,1}, \ldots, \delta_{i,k}\}$ be the set of deletion indices in the ith head of a d-head racetrack memory, i.e. $\boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t$, for $i \in [1, d-1]$. An interval $\mathcal{I}$ is* deletion isolated *if*

$$\boldsymbol{\delta}_{i+1} \cap \mathcal{I} = t + \boldsymbol{\delta}_i \cap \mathcal{I},$$

for $i \in [1, d-1]$.

Example 4. *Consider a 3-head racetrack memory with adjacent head distance $t = 2$. Let the length of the sequence $\mathbf{c}$ be $n = 22$ and let the deletion positions in three heads be given by*

$$\boldsymbol{\delta}_1 = \{1, 2, 4, 9, 14, 17\},$$
$$\boldsymbol{\delta}_2 = \{3, 4, 6, 11, 16, 19\}, \text{ and}$$
$$\boldsymbol{\delta}_3 = \{5, 6, 8, 13, 18, 21\},$$

Then the intervals $[1, 8]$, $[9, 13]$, and $[14, 22]$ are all deletion isolated.

Intuitively, an interval $\mathcal{I}$ is deletion isolated when the subsequences $\mathbf{c}_{\mathcal{I}\cap\boldsymbol{\delta}_i^c}$ for $i \in [1, d]$ can be regarded as the d reads of a d-head racetrack memory with input $\mathbf{c}_{\mathcal{I}}$ after $|\boldsymbol{\delta}_1 \cap \mathcal{I}|$ deletions in each head.

Lemma 7. *(Proofs appear in Section IV.) For any positive integers n and $R \geq k+1$, let $\mathbf{c} \in \{0,1\}^{n+R}$ be a sequence such that $L(\mathbf{c}_{[1,n+k+1]}, \leq k) \leq 3k + \lceil \log n \rceil + 2 \triangleq T$. Let the adjacent head distance t satisfy $t \geq (4k+1)(T+2k+1)$. Then, given $\boldsymbol{D} \in \mathcal{D}_k(\mathbf{c}, t)$, it is possible to find a set of $J \leq k$ disjoint and deletion isolated intervals $\mathcal{I}_j \subseteq [1, n+R]$, $j \in [i, J]$ such that $\boldsymbol{\delta}_w \subset \cup_{j=1}^{J} \mathcal{I}_j$ for $w \in [1, d]$ and*

$$\begin{aligned} &|\mathcal{I}_j \cap [1, n+k+1]| \\ \leq &(2\lfloor(2t+T+1)/2\rfloor + 1)kd + \lfloor(2t+T+1)/2\rfloor + k \\ \triangleq &B, \end{aligned} \tag{3}$$

for $j \in [1, J]$. Moreover, $|\boldsymbol{\delta}_1 \cap \mathcal{I}_j|$ can be determined for $j \in [1, J]$.

An illustration of Lemma 7 is shown in Fig. 2. Since the interval $\mathcal{I}_j$ is deletion isolated for $j \in [1, J]$, all rows of $\boldsymbol{D}$ are aligned in indices $[1, n+R] \backslash (\cup_{j=1}^{J} \mathcal{I}_j)$, i.e., the entries in the ith column of $\boldsymbol{D}$ correspond to the same bit in $\mathbf{c}$ for $i \in [1, n+R] \backslash (\cup_{j=1}^{J} \mathcal{I}_j)$. Let $\mathbf{c} \in \{0,1\}^{n+R}$ be a sequence satisfying $L(\mathbf{c}_{[1,n+k+1]}, \leq k) \leq T$. By virtue of Lemma 7, the bit c_i can be determined by

$$c_i = \boldsymbol{D}_{1, i - \sum_{j:\mathcal{I}_j \subseteq [1, i-1]} |\boldsymbol{\delta}_1 \cap \mathcal{I}_j|} \tag{4}$$

for $i \in [1, n+k+1] \backslash (\cup_{j=1}^{J} \mathcal{I}_j)$. In addition, let $\mathcal{I}_j = [b_j^{min}, b_j^{max}]$ for $j \in [1, J]$ such that $b_{j-1}^{max} < b_j^{min}$ for $j \in [2, J]$. Since $\mathcal{I}_j$ is deletion isolated for $j \in [1, J]$, the submatrix

$$\begin{aligned} &\boldsymbol{D}_{[1,d],[b_j^{min} - \sum_{i=1}^{j-1} |\boldsymbol{\delta}_1 \cap \mathcal{I}_i|, b_j^{max} - \sum_{i=1}^{j} |\boldsymbol{\delta}_1 \cap \mathcal{I}_i|]} \\ \in &\mathcal{D}_{|\boldsymbol{\delta}_1 \cap [b_j^{min}, b_j^{max}]|}(\mathbf{c}_{\mathcal{I}_j}, t) \end{aligned}$$

can be regarded as the d reads of the d-head racetrack memory with input $\mathbf{c}_{\mathcal{I}_j}$. According to Lemma 6, the bits $\mathbf{c}_{\mathcal{I}_j}$ with $|\boldsymbol{\delta}_1 \cap \mathcal{I}_j| < d$ can be recovered from

$$\boldsymbol{D}_{[1,d],[b_j^{min} - \sum_{i=j+1}^{J} |\boldsymbol{\delta}_1 \cap \mathcal{I}_i|, b_j^{max} - \sum_{i=j}^{J} |\boldsymbol{\delta}_1 \cap \mathcal{I}_i|]}$$

if the adjacent head distance satisfies $t \geq T[k(k-1)/2 + 1] + (7k - k^3)/6$. Note that there is at most a single interval $\mathcal{I}_{j_1}$ satisfying $|\boldsymbol{\delta}_1 \cap \mathcal{I}_{j_1}| \geq d$ when $k \leq 2d-1$. Hence, we are left to recover interval $\mathcal{I}_{j_1}$.

Split $\mathbf{c}_{[1,n+k+1]}$ into blocks

$$\mathbf{a}_i = \mathbf{c}_{[(i-1)B+1, \min\{iB, n+k+1\}]}, \ i \in [1, \lceil (n+k+1)/B \rceil] \tag{5}$$

of length B (defined in (3)) except that $\mathbf{a}_{\lceil (n+k+1)/B \rceil}$ may have length shorter than B. Since $|\mathcal{I}_{j_1} \cap [1, n+k+1]| \leq B$, the interval $\mathcal{I}_{j_1}$ spans over at most two blocks $\mathbf{a}_{j_1'}$ and $\mathbf{a}_{j_1'+1}$. It then follows that there are at most two consecutive blocks, where $\mathcal{I}_{j_1}$ lies in, that remain to be recovered. Moreover, at most k deletions occur in interval $\mathcal{I}_{j_1}$, and hence in blocks $\mathbf{a}_{j_1'}$ and $\mathbf{a}_{j_1'+1}$.

For a positive integer n and a sequence $\mathbf{c} \in \{0,1\}^{n+k+1}$ of length $n+k+1$, let the mapping $S : \{0,1\}^{n+k+1} \to \mathbb{F}_{4k\log B+o(\log B)}^{\lceil(n+k+1)/B\rceil}$ be defined by

$$S(\mathbf{c}) = (H(\mathbf{a}_1), H(\mathbf{a}_2), \ldots, H(\mathbf{a}_{\lceil(n+k+1)/B\rceil})), \tag{6}$$

where $\mathbf{a}_i$, $i \in [1, \lceil(n+k+1)/B\rceil]$ are the blocks of $\mathbf{c}$ defined in Eq. (5). The mapping $H(\mathbf{a}_{\lceil(n+k+1)/B\rceil})$, given in Lemma 3, takes the input $\mathbf{a}_{\lceil(n+k+1)/B\rceil}$ of length at most B. The sequence $S(\mathbf{c})$ is a concatenation of the mappings H of blocks of $\mathbf{c}$.

Lemma 8. *For constant positive integers k, d, and B defined in (3), there exists a function*

$$\begin{aligned} DecS : &\{0,1\}^{n+1} \times \{0,1\}^{\lceil(n+k+1)/B\rceil(4k\log B+o(\log B))} \\ &\to \{0,1\}^{n+k+1}, \end{aligned}$$

such that for any sequence $\mathbf{c} \in \{0,1\}^{n+k+1}$ and its length $n+1$ subsequence $\mathbf{d} \in \{0,1\}^{n+1}$, we have that $DecS(\mathbf{d}, S(\mathbf{c})) = \mathbf{c}$, i.e., the sequence $\mathbf{c}$ can be recovered from k deletions with the help of $S(\mathbf{c})$.

Proof. Note that $\mathbf{d}_{[(i-1)B+1,\min\{iB,n+k+1\}-k]}$ is a length $B-k$ subsequence of the block $\mathbf{a}_i$ for $i \in \{1, \ldots, \lceil(n+k+1)/B\rceil\}$. According to Lemma 3, the block $\mathbf{a}_i$ can be recovered from $\mathbf{d}_{(i-1)B+1,\max\{iB,n+k+1\}-k}$ with the help of $H(\mathbf{a}_i)$. Thus the sequence $\mathbf{c}$ can be recovered. □

We are now ready to present the code construction. For any sequence $\mathbf{c} \in \{0,1\}^n$, define the following encoding function:

$$Enc_1(\mathbf{c}) = (F(\mathbf{c}), R_1^{'}(\mathbf{c}), R_1^{''}(\mathbf{c})) \tag{7}$$

where

$$\begin{aligned} R_1^{'}(\mathbf{c}) &= ER(S(F(\mathbf{c}))), \\ R_1^{''}(\mathbf{c}) &= Rep_{k+1}(H(R_1'(\mathbf{c}))), \end{aligned} \tag{8}$$

and the function Rep_{k+1} is a $k+1$-fold repetition function that repeats each bit $k+1$ times. The mapping ER is defined in Lemma 2 to correct two consecutive symbol erasures in $S(F(\mathbf{c}))$, and the mapping $F(\mathbf{c}) \in \{0,1\}^{n+k+1}$ is defined in Lemma 5, to obtain a sequence satisfying $L(F(\mathbf{c}), \leq k) \leq T$ so that Lemma 7 can be applied. The redundancy consists of two layers. The function $R_1^{'}(\mathbf{c})$ can be regarded as the first layer redundancy, with the help of which $F(\mathbf{c})$ can be recovered from k deletions. It computes the redundancy $ER(S(F(\mathbf{c})))$ that can be used to recover $S(F(\mathbf{c}))$. According to Lemma 8, the recovered $S(F(\mathbf{c}))$ can be then used to recover $F(\mathbf{c})$. The function $R_1^{''}(\mathbf{c})$ can be regarded as the second layer redundancy that helps recover itself and $R_1^{'}(\mathbf{c})$ from k deletions.

The length of $R_1^{'}(\mathbf{c})$ is given by $N_1 = 8k\log B + o(\log B) = 8k\log t + o(\log t)$. The length of $R_1^{''}(\mathbf{c})$ is $N_2 = 4k(k+1)\log N_1 + O(1) = o(\log t)$. The length of the codeword $Enc_1(\mathbf{c})$ is given by $N = n+k+1+N_1+N_2 = 8k\log t + o(\log t)$. The next theorem proves Theorem 1 for cases when $d \leq k \leq 2d-1$.

Theorem 3. *The set $\mathcal{C}_1 = \{Enc_1(\mathbf{c}) : \mathbf{c} \in \{0,1\}^n\}$ is a d-head k-deletion correcting code for $d \leq k \leq 2d-1$, if the adjacent head distance t satisfies $t \geq \max\{(3k + \lceil\log n\rceil + 2)[k(k-1)/2+1] + (7k-k^3)/6, (4k+1)(5k+\lceil\log n\rceil+3)\}$, $i \in [1, d-1]$. The code $\mathcal{C}_1$ can be constructed, encoded, and decoded in $O(nt^{2k+1} + poly(n))$ time. The redundancy of $\mathcal{C}_1$ is $N - n = 8k\log t + o(\log t)$.*

Proof. For any $\boldsymbol{D} \in \mathcal{D}_k(\mathbf{c}, t)$, let $\mathbf{d} = \boldsymbol{D}_{1,[1,N-k]}$ be the first row of $\boldsymbol{D}$, i.e., the first read. The sequence $\mathbf{d}$ is a length $N-k$ subsequence of $Enc_1(\mathbf{c})$. We first show how to recover $R_1^{'}(\mathbf{c})$ from $\mathbf{d}$. Note that $\mathbf{d}_{[N-N_2+1,N-k]}$ is a length N_2-k subsequence of $R_1^{''}(\mathbf{c})$, the $k+1$-fold repetition of $H(R_1^{'}(\mathbf{c}))$. Since a $k+1$-fold repetition code is a k-deletion code, the mapping $H(R_1^{'}(\mathbf{c}))$ can be recovered. Furthermore, we have that $\mathbf{d}_{[n+k+2,n+k+1+N_1-k]}$ is a length N_1-k subsequence of $R_1^{'}(\mathbf{c})$. Hence according to Lemma 3, we can obtain $R_1^{'}(\mathbf{c})$ from $\mathbf{d}_{[n+k+2,n+k+1+N_1-k]}$, with the help of $H(R_1^{'}(\mathbf{c}))$.

Next, we show how to use $R_1^{'}(\mathbf{c})$ to recover $F(\mathbf{c})$. Note that $L(F(\mathbf{c}), \leq k) \leq T$. From Lemma 7 and the discussion that follows, we can separate $F(\mathbf{c})$ into blocks $\mathbf{a}_i$, $i \in [1, \lceil(n+K+1)/B\rceil]$, of length B, and identify a set of deletion isolated intervals $\{\mathcal{I}_j\}_{j=1}^J$ that contain all the deletion indices in all reads. We then use (4) to recover the bits with indices outside the intervals $\{\mathcal{I}_j\}_{j=1}^J$. Then, we apply Lemma 6 to correct the errors in intervals $\mathcal{I}_j$ such that $|\boldsymbol{\delta}_1 \cap \mathcal{I}_j| < d$. Note that there is at most one interval $\mathcal{I}_{j'}$ with at least d deletions, where j' can be determined by looking for the unique interval $\mathcal{I}_{j'}$ such that $|\mathcal{I}_{j'} \cap \boldsymbol{\delta}_1| \geq d$ (Note that $|\mathcal{I}_{j'} \cap \boldsymbol{\delta}_1|$ is known by Lemma 7.). The interval $\mathcal{I}_{j'}$ covers at most two consecutive blocks $\mathbf{a}_{j_1}$ and $\mathbf{a}_{j_1+1}$, where the index j_1 is known. This implies that $S(F(\mathbf{c}))$ can be retrieved with consecutive at most two symbol erasures, the position of which can be identified. Hence, we can use $R_1^{'}(\mathbf{c})$ to recover $S(F(\mathbf{c}))$ and find the mapping $H(\mathbf{a}_{j_1})$ and $H(\mathbf{a}_{j_1+1})$, since $R_1^{'}(\mathbf{c}) = ER(S(F(\mathbf{c})))$ corrects the two consecutive symbol errors in $S(F(\mathbf{c}))$ by Lemma 2. Note that $D_{1,[1,n+1]}$ is a length $n+1$ subsequences of $F(\mathbf{c})$. Hence from Lemma 8 the sequence $F(\mathbf{c})$ and thus $\mathbf{c}$ can be recovered given $S(F(\mathbf{c}))$. The computation of $S(F(\mathbf{c}))$, which computes $O(n/B)$ times the mapping $H(\mathbf{a}_i)$, $[1, \lceil(n+k+1)/B\rceil]$, constitutes the main part of the computation complexity of $Enc_1(\mathbf{c})$. Since the computation of $H(\mathbf{a}_i)$ takes $O(B^{2k+1}) = O(t^{2k+1})$ time for each $i \in [1, \lceil(n+K+1)/B\rceil]$ and computing $F(\mathbf{c})$ requires $O(poly(n))$ time, it takes $O(nt^{2k+1} + poly(n))$ time to compute $Enc_1(\mathbf{c})$. □

IV. Proof of Lemma 7

Let $\boldsymbol{D} \in \mathcal{D}_k(\mathbf{c}, t)$ be the d reads from all heads, where $\mathbf{c} \in \{0,1\}^{n+R}$ satisfies $L(\mathbf{c}_{[1,n+k+1]}, \leq k) \leq T$. Then $\boldsymbol{D}$ is a d by $n+R-k$ matrix. The proof of Lemma 7 consists of two steps. The first step is to identify a set of disjoint intervals $\mathcal{I}_j^*$, $j \in [1, J]$ that satisfy

(P1) There exist a set of disjoint and deletion isolated intervals $\mathcal{I}_j$, $j \in [1, J]$, such that $\boldsymbol{D}_{w,\mathcal{I}_j^*} = \mathbf{c}_{\mathcal{I}_j \cap \boldsymbol{\delta}_w^c}$ for $w \in [1, d]$ and $j \in [1, J]$, i.e., the subsequence $\boldsymbol{D}_{w,\mathcal{I}_j^*}$ comes from $\mathbf{c}_{\mathcal{I}_j}$ in the wth read after deleting $\mathbf{c}_{\mathcal{I}_j \cap \boldsymbol{\delta}_w}$.

(P2) $J \leq k$ and $\boldsymbol{\delta}_w \subseteq \cup_{j=1}^J \mathcal{I}_j$ for $w \in [1, d]$.

(P3) $|\mathcal{I}_j^* \cap [1, n+1]| \leq B - k$

The algorithm for finding intervals $\mathcal{I}_j^*$, $j \in [1,J]$ is given in Algorithm 2. The second step is to determine the number of deletions $|\boldsymbol{\delta}_w \cap \mathcal{I}_j|$ for $w \in [1,d]$ and $j \in [1,J]$, that happen in each interval in each head, based on $\boldsymbol{D}_{[1,d],\mathcal{I}_j^*}$. The algorithm for determining $|\boldsymbol{\delta}_w \cap \mathcal{I}_j|$, $w \in [1,d]$ and $j \in [1,J]$ is given in Algorithm 3. Then we have that

$$\mathcal{I}_j = [i_{2j-1} + \sum_{\ell=1}^{j-1} |\boldsymbol{\delta}_1 \cap \mathcal{I}_\ell|, i_{2j} + \sum_{\ell=1}^{j} |\boldsymbol{\delta}_1 \cap \mathcal{I}_\ell|], \qquad (9)$$

where i_{2j-1} and i_{2j} are the starting and ending points of the interval $\mathcal{I}_j^*$. It is assumed that $i_j > i_l$ for $j > l$. The disjointness of $\mathcal{I}_j$, $j \in [1,J]$ follows from the fact that $\mathcal{I}_j^*$, $j \in [1,J]$ are disjoint. The algorithm for finding the intervals $\mathcal{I}_j$, $j \in [1,J]$, for Lemma 7 is summarized in Algorithm 1.

Algorithm 1: Finding intervals $\mathcal{I}_j$, $j \in [1,J]$

```
**Input:** The read matrix $\boldsymbol{D} \in \mathcal{D}_k(\mathbf{c},t)$. **Output:** The
        intervals $\mathcal{I}_j$, $j \in [1,J]$, described in Lemma 7
**Step 1:** Apply Algorithm 2 to obtain intervals
        $\mathcal{I}_j^* = [i_{2j-1}, i_{2j}]$, $j \in [1,J]$. Go to Step 2;
**Step 2:** Apply Algorithm 3 to compute
    $|\boldsymbol{\delta}_w \cap \mathcal{I}_j|$, $w \in [1,d]$ and $j \in [1,J]$. Go to Step 3;
**Step 3:** Output the intervals $\mathcal{I}_j$, $j \in [1,J]$, computed by
    (9);
```

In the following two subsections, we present the details of Algorithm 2 and Algorithm 3, respectively.

Algorithm 2: Finding Intervals $\{\mathcal{I}_j^*\}_{j=1}^J$

```
**Input:** The read matrix $\boldsymbol{D} \in \mathcal{D}_k(\mathbf{c},t)$. **Output:** Intervals
        $\{\mathcal{I}_j^*\}_{j=1}^J$ satisfying properties **(P1), (P2), and**
        **(P3)**
**Initialization:** Set all integers $m \in [1, n+R-k]$
        unmarked. Let $i = 1$. Find the largest positive
        integer $L$ such that the sequences $\boldsymbol{D}_{w,[i,i+L-1]}$
        are equal for all $w \in [1,d]$. If such $L$ exists and
        satisfies $L > t$, mark the integers $m \in [1, L-t]$
        and go to Step 1. Otherwise, go to Step 1;
**Step 1:** Find the largest positive integer $L$ such that the
    sequences $\boldsymbol{D}_{w,[i,i+L-1]}$ are equal for all $w \in [1,d]$. Go
    to Step 2. If no such $L$ is found, set $L = 0$ and go to
    Step 2;
**Step 2:** If $L \geq 2t+T+1$, mark the integers
    $m \in [i+t, \min\{i+L-1, n+1\} - t]$. Set $i = i+L$ and
    go to Step 3. Else $i = i+1$ and go to Step 3;
**Step 3:** If $i \leq n+1$, go to Step 1. Else go to Step 4;
**Step 4:** If the number of unmarked intervals (An
    unmarked interval $[i,j]$ means that $m \in [i,j]$ are not
    marked and $i-1$ and $j+1$ are marked. It is assumed
    that 0 and $n+R-k+1$ are marked.) within $[1, n+1]$ is
    not greater than $k$, output all unmarked intervals. Else
    output the first $k$ intervals, i.e., the intervals with the
    minimum $k$ starting indices;
```

A. Identifying Intervals $\mathcal{I}_j^*$

We prove that the output intervals $\mathcal{I}_1^*, \ldots, \mathcal{I}_J^*$ of Algorithm 2 satisfy the above constraints **(P1)**, **(P2)**, and **(P3)**. The following lemma will be used.

Lemma 9. *Let $\boldsymbol{D} \in \mathcal{D}_k(\mathbf{c},t)$ for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}_{[1,n+k+1]}, \leq k) \leq T$. Let the adjacent head distance t satisfy $t \geq k(T+1)+1$. If the sequences $\boldsymbol{D}_{w,[i_1,i_2]}$ are equal for all $w \in [1,d]$ in some interval $[i_1,i_2] \subseteq [1,n+1]$ with length $i_2 - i_1 + 1 \geq 2t+T+1$, then no deletions occur within bits $\boldsymbol{D}_{w,[i_1+t,i_2-t]}$ for all w, i.e., there exist integers $i_1' = i_1 + t + |\boldsymbol{\delta}_j \cap [1, i_1'-1]|$ and $i_2' = i_2 - t + |\boldsymbol{\delta}_j \cap [1, i_2'-1]|$, such that $\mathbf{c}_{[i_1',i_2']} = \boldsymbol{D}_{w,[i_1+t,i_2-t]}$ and $[i_1',i_2'] \cap \boldsymbol{\delta}_w = \emptyset$ for $w \in [1,d]$. In addition, both intervals $[1, i_1'-1]$ and $[i_2'+1, n+R]$ are deletion isolated.*

Proof. Let $c_{i_0'}$, $c_{i_1'}$, $c_{i_2'}$, and $c_{i_3'}$ be the bits that become $\boldsymbol{D}_{1,i_1}$, $\boldsymbol{D}_{1,i_1+t}$, $\boldsymbol{D}_{1,i_2-t}$, and $\boldsymbol{D}_{1,i_2}$ respectively after deletions, i.e., $i_0' - |\boldsymbol{\delta}_1 \cap [1, i_0'-1]| = i_1$, $i_1' - |\boldsymbol{\delta}_1 \cap [1, i_1'-1]| = i_1 + t$, $i_2' - |\boldsymbol{\delta}_1 \cap [1, i_2'-1]| = i_2 - t$, and $i_3' - |\boldsymbol{\delta}_1 \cap [1, i_3'-1]| = i_2$. We show that no deletions occur within $\boldsymbol{D}_{w,[i_1,i_2-t]}$ for $w \in [1,d-1]$ or within $\boldsymbol{D}_{w,[i_1+t,i_2]}$ for $w \in [2,d]$, i.e., $\boldsymbol{\delta}_w \cap [i_0', i_2'] = \emptyset$ for $w \in [1,d-1]$, and $\boldsymbol{\delta}_w \cap [i_1', i_3'] = \emptyset$ for $w \in [2,d]$.

Suppose on the contrary, there are deletions within $\boldsymbol{D}_{w,[i_1,i_2-t]}$ for $w \in [1,d-1]$. Then there exist some $w_1 \in [1,d-1]$ and $k_1 \in [1,k]$, such that $\delta_{w_1,k_1} \in [i_0', i_2']$ (recall that δ_{w_1,k_1} is the index of the k_1th deletion in the w_1th read). Then we have that $\delta_{w_1+1,k_1} = \delta_{w_1,k_1} + t \in [i_0', i_3']$. Note that there are $k - k_1$ deletions $\{\delta_{w_1,k_1+1}, \ldots, \delta_{w_1,k}\}$ to the right of δ_{w_1,k_1} and $k_1 - 1$ deletions $\{\delta_{w_1+1,1}, \ldots, \delta_{w_1+1,k_1-1}\}$ to the left of δ_{w_1+1,k_1}. Hence we have that

$$\begin{aligned}
&|(\boldsymbol{\delta}_{w_1} \cup \boldsymbol{\delta}_{w_1+1}) \cap [\delta_{w_1,k_1}+1, \delta_{w_1,k_1}+t-1]| \\
\leq &|(\boldsymbol{\delta}_{w_1} \cup \boldsymbol{\delta}_{w_1+1}) \cap [\delta_{w_1,k_1}+1, \delta_{w_1+1,k_1}-1]| \\
\leq &k - k_1 + k_1 - 1 \\
= &k - 1,
\end{aligned}$$

meaning that there are at most $k-1$ deletions in the w_1th or (w_1+1)th heads that lie in interval $[\delta_{w_1,k_1}+1, \delta_{w_1,k_1}+t-1]$. Since $t \geq k(T+1)+1$, there are at least k disjoint intervals of length $T+1$ that lie in interval $[\delta_{w_1,k_1}+1, \delta_{w_1,k_1}+t-1]$. It then follows that there exists an interval $[i', i'+T] \subset [\delta_{w_1,k_1}+1, \delta_{w_1,k_1}+t-1]$ such that $[i', i'+T] \cap (\boldsymbol{\delta}_{w_1} \cup \boldsymbol{\delta}_{w_1+1}) = \emptyset$. Let $l_1' = |\boldsymbol{\delta}_{w_1} \cap [1, i'-1]|$ and $l_2' = |\boldsymbol{\delta}_{w_1+1} \cap [1, i'-1]|$ be the number of deletions in heads w_1 and w_1+1, respectively that is to the left of i'. We have that $l_1' > l_2'$ since $\delta_{w_1,k_1} < i'$ and $\delta_{w_1+1,k_1} > i'+T$. Since $[i', i'+T] \cap (\boldsymbol{\delta}_{w_1} \cup \boldsymbol{\delta}_{w_1+1}) = \emptyset$ and $l_1' - l_2' \leq k < T$, we have that

$$\begin{aligned}
l_1' = &|\boldsymbol{\delta}_{w_1} \cap [1, i'-1]| \\
= &|\boldsymbol{\delta}_{w_1} \cap [1, i'+l_1'-l_2'-1]| \\
= &|\boldsymbol{\delta}_{w_1} \cap [1, i'+T-1]|, \text{ and} \\
l_2' = &|\boldsymbol{\delta}_{w_1+1} \cap [1, i'-1]| \\
= &|\boldsymbol{\delta}_{w_1+1} \cap [1, i'+T+l_2'-l_1'-1]|.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\mathbf{c}_{[i'+l'_1-l'_2,i'+T]}\\
=&\boldsymbol{D}_{w_1,[i'+l'_1-l'_2-|\boldsymbol{\delta}_{w_1}\cap[1,i'+l'_1-l'_2-1]|,i'+T-|\boldsymbol{\delta}_{w_1}\cap[1,i'+T-1]|]}\\
=&\boldsymbol{D}_{w_1,[i'-l'_2,i'+T-l'_1]}\\
=&\boldsymbol{D}_{w_1+1,[i'-l'_2,i'+T-l'_1]}\\
=&\boldsymbol{D}_{w_1+1,[i'-|\boldsymbol{\delta}_{w_1+1}\cap[1,i'-1]|,i'+T+l'_2-l'_1-|\boldsymbol{\delta}_{w_1+1}\cap[1,i'+T+l'_2-l'_1-1]|]}\\
=&\mathbf{c}_{[i',i'+T+l'_2-l'_1]},
\end{aligned}$$

which implies that $L(\mathbf{c}_{[i',i'+T]},l'_1-l'_2)=T+1>T$. Since $[i',i'+T]\subset[i'_0,i'_3]\subset[1,n+k+1]$, this is a contradiction to the assumption that $L(\mathbf{c}_{[1,n+k+1]},l'_1-l'_2)\leq T$. Therefore, there are no deletions within $\boldsymbol{D}_{w,[i_1,i_2-t]}$ for $w\in[1,d-1]$, i.e., $\boldsymbol{\delta}_w\cap[i'_0,i'_2]=\emptyset$ for $w\in[1,d-1]$. Similarly, we have that $\boldsymbol{\delta}_w\cap[i'_1,i'_3]=\emptyset$ for $w\in[2,d]$. Since $[i'_1,i'_2]\subset[i'_0,i'_2]$ and $[i'_1,i'_2]\subset[i'_1,i'_3]$, it follows that

$$[i'_1,i'_2]\cap\boldsymbol{\delta}_w=\emptyset \tag{10}$$

and hence $\mathbf{c}_{[i'_1,i'_2]}=\boldsymbol{D}_{w,[i_1+t,i_2-t]}$ for $w\in[1,d]$.

Next we show that the intervals $[1,i'_1-1]$ and $[i'_2+1,n+R]$ are deletion isolated. Suppose on the contrary, there exists some $w_2\in[1,d]$ for which $(\boldsymbol{\delta}_{w_2}\cap[1,i'_1-1])+t\neq(\boldsymbol{\delta}_{w_2+1}\cap[1,i'_1-1])$. Then we have that $|\boldsymbol{\delta}_{w_2}\cap[1,i'_1-1]|>|\boldsymbol{\delta}_{w_2+1}\cap[1,i'_1-1]|$. Let $x=|\boldsymbol{\delta}_{w_2}\cap[1,i'_1-1]|-|\boldsymbol{\delta}_{w_2+1}\cap[1,i'_1-1]|$, then,

$$\begin{aligned}
&\mathbf{c}_{[i'_1,i'_2-x]}\\
\overset{(a)}{=}&\boldsymbol{D}_{w_2,[i_1+t+|\boldsymbol{\delta}_{w_2}\cap[1,i'_1-1]|,i_2-t-x+|\boldsymbol{\delta}_{w_2}\cap[1,i'_2-x-1]|]}\\
=&\boldsymbol{D}_{w_2+1,[i_1+t+|\boldsymbol{\delta}_{w_2}\cap[1,i'_1-1]|,i_2--x+|\boldsymbol{\delta}_{w_2}\cap[1,i'_2-x-1]|]}\\
=&\boldsymbol{D}_{w_2+1,[i_1+t+x+|\boldsymbol{\delta}_{w_2+1}\cap[1,i'_1-1]|,i_2-t+|\boldsymbol{\delta}_{w_2+1}\cap[1,i'_2-1]|]}\\
\overset{(b)}{=}&\mathbf{c}_{[i'_1+x,i'_2]},
\end{aligned} \tag{11}$$

where (a) and (b) hold since we have E.q. (10). This implies that

$$\begin{aligned}
L(\mathbf{c}_{i'_1,i'_2},x)=&i'_2-i'_1+1\\
\overset{(a)}{\geq}&i_2-i_1-2t+1\\
\geq&T+1,
\end{aligned}$$

where (a) holds since $\mathbf{c}_{[i'_1,i'_2]}=\boldsymbol{D}_{w,[i_1+t,i_2-t]}$. This contradicts to the fact that $L(\mathbf{c}_{[1,n+k-1],\leq k})\leq T$. Therefore, the interval $[1,i'_1-1]$ is deletion islolated. Similarly, $[i'_2+1,n+R]$ is deletion isolated. □

In the following, we show that the output intervals satisfy **(P1)**, **(P2)**, and **(P3)**, respectively. Let $[p_{2j-1},p_{2j}]$, $j\in[1,J']$ be the marked intervals in the algorithm, where $p_1<\ldots<p_{2J'}$. Let $p_0=0$ and $p_{2J'+1}=n+R+1-k$, then the output intervals are the leftmost up to k nonempty intervals among $\{[p_{2j}+1,p_{2j+1}-1]\}_{j=0}^{J'}$. Note that from the marking operation in the **Initialization** step and **Step 2** in Algorithm 2, the interval $[n+1-t,n+R-k]$ is not marked. In addition, for any $j\in[1,J']$, sequences $\boldsymbol{D}_{w,[p_{2j-1},p_{2j}]}$ are equal for all $w\in[1,d]$. Hence, according to Lemma 9, there exist intervals $[p'_{2j-1},p'_{2j}]$, $j\in[1,J']$, where

$$\begin{aligned}
&p'_j=p_j+|\boldsymbol{\delta}_w\cap[1,p'_j-1]|,\text{ and}\\
&[p'_{2\ell-1},p'_{2\ell}]\cap\boldsymbol{\delta}_w=\emptyset,
\end{aligned} \tag{12}$$

for all $j\in[1,2J']$, $\ell\in[1,J']$, and $w\in[1,d]$. In addition, intervals $[1,p'_{2j-1}-1]$ are deletion isolated[3] for $j\in[1,J']$. It follows that $[p'_{2j-1},p'_{2j+1}-1]$ is deletion isolated for $j\in[1,J']$, where $p'_{2J'+1}=n+R+1$. Since $[p'_{2j-1},p'_{2j}]\cap\boldsymbol{\delta}_w=\emptyset$ for $j\in[1,J']$ and $w\in[1,d]$, then we have that the intervals $[p'_{2j}+1,p'_{2j+1}-1]$, $j\in[0,J']$, where $p'_0=0$ and $p'_{2J+1}=n+R+1$, are deletion isolated. From (12) we have that $\boldsymbol{D}_{w,[p_{2j}+1,p_{2j+1}-1]}=\mathbf{c}_{[p'_{2j}+1,p'_{2j+1}-1]\cap\boldsymbol{\delta}^c_w}$. In addition, the intervals $\{[p'_{2j}+1,p'_{2j+1}-1]\}_{j=0}^{J'}$ are disjoint since

$$\begin{aligned}
&(p'_{2(j+1)}+1)-(p'_{2j+1}-1)\\
=&p_{2(j+1)}+|\boldsymbol{\delta}_w\cap[1,p'_{2(j+1)}-1]|+2\\
&-p_{2j+1}-|\boldsymbol{\delta}_w\cap[1,p'_{2j+1}-1]|\\
\overset{(a)}{\geq}&T+|\boldsymbol{\delta}_w\cap[1,p'_{2(j+1)}-1]|-|\boldsymbol{\delta}_w\cap[1,p'_{2j+1}-1]|\\
\geq&T-k>0,
\end{aligned}$$

for $j\in[0,J'-1]$, where (a) follows from the fact that marked intervals have length at least T. Therefore, the output intervals $\{[p_{2j}+1,p_{2j+1}-1]\}_{j=0}^{J'}$ satisfy **(P1)**.

Next, we show that the output intervals satisfy **(P2)**. For any output interval $[p_{2j}+1,p_{2j+1}-1]$ with $[p_{2j}+1,p_{2j+1}-1]\subseteq[1,n+1-t]$, the corresponding interval $[p'_{2j}+1,p'_{2j+1}-1]$ contains at least one deletion in $\boldsymbol{\delta}_w$, i.e., $[p'_{2j}+1,p'_{2j+1}-1]\cap\boldsymbol{\delta}_w\neq\emptyset$, for some $w\in[1,d]$. Otherwise, we have that $[p'_{2j'}+1,p'_{2j'+1}-1]\cap\boldsymbol{\delta}_w=\emptyset$ for $w\in[1,d]$ for some j'. Combining with (12) and the fact that intervals $[1,p'_{2j-1}-1]$ are deletion isolate for $j\in[1,J']$, it follows that the sequences $\boldsymbol{D}_{w,[p_{2j'}+1,p_{2j'+1}-1]}$ are equal for $w\in[1,d]$. This implies that the interval $[p_{2j'}+1,p_{2j'+1}-1]$ is marked during the procedure, which is a contradiction to the fact that $[p_{2j'}+1,p_{2j'+1}-1]$ is not marked. Therefore, there are at most k unmarked intervals that lie within the interval $[1,n+1]$. Note that there is one unmarked interval containing $[n+1-t,n+R-k]$ that does not lie in $[1,n+1]$. It follows that there are at most $k+1$ unmarked intervals in total. When there are $k+1$ unmarked intervals, the deletions $\boldsymbol{\delta}_w$ are contained in the k output intervals since each output interval within $[1,n+1]$ contains at least one deletion. When there are no more than k intervals, the deletions are contained in the unmarked output intervals since the marked intervals do not contain deletions. Therefore we have that $\boldsymbol{\delta}_w\subseteq\{[p_{2j}+1,p_{2j+1}-1]\}_{j=1}^{J}$, where $\{[p_{2j}+1,p_{2j+1}-1]\}_{j=1}^{J}$ are the output intervals and $J\leq k$.

Finally, we show that $|\mathcal{I}^*_j\cap[1,n+1]|\leq B-k$ for $j\in[1,J]$, which is **(P3)**. We first prove that for any unmarked index $i\in[1,n+1-\lfloor t+(T+1)/2\rfloor]$, there exist some $w\in[1,d]$ and $k_1\in$

[3]The interval $[p_1,p_2]$ may be marked in the **Initialization** step in Algorithm 2 and have length less than $T+2t+1$. In that case, apply Lemma 9 by considering an interval $[-t+T+1,0]$ where $\boldsymbol{D}_{w,[-t+T+1,0]}$ are equal for $w\in[1,d]$.

$[1,k]$, such that a deletion at δ_{w,k_1} occurs within distance $\lfloor t+(T+1)/2\rfloor$ to the bit $\mathbf{c}_{i'=i+|\boldsymbol{\delta}_w\cap[1,i'-1]|}$ that becomes $\boldsymbol{D}_{w,i}$, i.e., $\delta_{w,k_1}\in[i'-\lfloor t+(T+1)/2\rfloor, i'+\lfloor t+(T+1)/2\rfloor]$[4]. Otherwise, we have that $[i'-\lfloor t+(T+1)/2\rfloor, i'+\lfloor t+(T+1)/2\rfloor]\cap\delta_w=\emptyset$ for $w\in[1,d]$. Since $[i'-\lfloor t+(T+1)/2\rfloor, i'+\lfloor t+(T+1)/2\rfloor]$ has length more than t for $w\in[1,d]$, we have that $\delta_{w+1,j}=\delta_{w,j}+t\in[1,i'-\lfloor t+(T+1)/2\rfloor-1]$ for every $\delta_{w,j}+t\in[1,i'-\lfloor t+(T+1)/2\rfloor-1]$. It follows that $[1,i'-\lfloor t+(T+1)/2\rfloor-1]$ is deletion isolated. Therefore, we have that

$$\begin{aligned}&\boldsymbol{D}_{w,[i-\lfloor t+(T+1)/2\rfloor, i+\lfloor t+(T+1)/2\rfloor]}\\ =&\mathbf{c}_{[i-\lfloor t+(T+1)/2\rfloor+|\boldsymbol{\delta}_w\cap[i'-1]|, i+\lfloor t+(T+1)/2\rfloor+|\boldsymbol{\delta}_w\cap[i'-1]|]}\\ =&\mathbf{c}_{[i'-\lfloor t+(T+1)/2\rfloor, i'+\lfloor t+(T+1)/2\rfloor]}\end{aligned}$$

are equal for all $w\in[1,d]$, which means that the interval $[i-\lfloor t+(T+1)/2\rfloor, i+\lfloor t+(T+1)/2\rfloor]$ and thus the index i should be marked. Therefore, every unmarked index $i\in[1,n+1-\lfloor t+(T+1)/2\rfloor]$ is associated with a deletion index δ_{w,k_1} that is within distance $\lfloor t+(T+1)/2\rfloor$ to $i'=i+|\boldsymbol{\delta}_w\cap[1,i'-1]|$. On the other hand, any deletion δ_{w,k_1} is associated with at most $2\lfloor(2t+T+1)/2\rfloor+1$ unmarked indices. Therefore, the number of unmarked bits within $[1,n+1-\lfloor t+(T+1)/2\rfloor]$ is at most $(2\lfloor(2t+T+1)/2\rfloor+1)kd$. The number of unmarked bits within $[1,n+1]$ is at most $(2\lfloor(2t+T+1)/2\rfloor+1)kd+\lfloor(2t+T+1)/2\rfloor=B-k$.

B. Determining the Number of Deletions

In this subsection we present Algorithm 3 for determining the number of deletions $|\boldsymbol{\delta}_w\cap\mathcal{I}_j|$, $w\in[1,d]$, for any deletion isolated interval $\mathcal{I}_j\subseteq[1,n+k+1]$. Fix j. The input for this algorithm are the reads $\boldsymbol{D}_{[1,d],\mathcal{I}^*_j}$ obtained by deleting $\mathbf{c}_{\boldsymbol{\delta}_w\cap\mathcal{I}_j}$, $w\in[1,d]$ from $\mathbf{c}_{\mathcal{I}_j}$. The interval $\mathcal{I}^*_j$ is the jth output interval obtained from Algorithm 2. Note that $\mathcal{I}_j$ is not known at this point. In the algorithm only the first two reads $\boldsymbol{D}_{[1,2],\mathcal{I}^*_j}$ are used. Let $\mathcal{I}_j=[b_{min},b_{max}]$ for some positive integers b_{min} and b_{max}. Consider the following intervals,

$$\begin{aligned}\mathcal{B}_{i,m}=&[b_{min}+(i-1)t+(m-1)(T+2k+1),\\ &\min\{b_{min}+(i-1)t+m(T+2k+1)-1,b_{max}\}],\\ &\text{for } i\in[1,\lceil(b_{max}-b_{min}+1)/t\rceil] \text{ and } m\in[1,\min\{4k+1,\\ &\lceil((b_{max}-b_{min}+1) \bmod t)/(T+2k+1)\rceil\}]\end{aligned}$$

The intervals $\mathcal{B}_{i,m}$ are disjoint and have length $T+2k+1$ except when $i=\lceil(b_{max}-b_{min}+1)/t\rceil$ and $m=\min\lceil((b_{max}-b_{min}+1) \bmod t)/(T+2k+1)\rceil$ the length might be less. Let $\mathcal{U}_m=\cup_i\mathcal{B}_{i,m}$ be the union of intervals $\mathcal{B}_{i,m}$ with the same m for $m\in[1,4k+1]$. Then the unions U_m are disjoint since $t\geq(4k+1)(T+2k+1)$. Since the deletions occur in at most $2k$ positions in the first two heads, at least $2k+1$ unions $\{\mathcal{U}_{m_1},\ldots,\mathcal{U}_{m_{2k+1}}\}$ satisfy $\mathcal{U}_{m_l}\cap(\boldsymbol{\delta}_1\cup\boldsymbol{\delta}_2)=\emptyset$ for $l\in[1,2k+1]$.

[4]When $i'-\lfloor t+(T+1)/2\rfloor<0$, consider bits $\boldsymbol{D}_{w,[i'-\lfloor t+(T+1)/2\rfloor,0]}$ that are equal for $w\in[1,d]$

Similarly, let $\mathcal{I}^*_j=[b'_{min},b'_{max}]$ for some positive integers b'_{min} and b'_{max}. Define the intervals

$$\begin{aligned}\mathcal{B}'_{i,m}=&[b'_{min}+(i-1)t+(m-1)(T+2k+1),\\ &\min\{b'_{min}+(i-1)t+m(T+2k+1)-k-1,b'_{max}\}],\\ &\text{for } i\in[1,\lceil(b'_{max}-b'_{min}+1)/t\rceil] \text{ and } m\in[1,\min\{4k+1,\\ &\lceil((b'_{max}-b'_{min}+1) \bmod t)/(T+2k+1)\rceil\}]\end{aligned}\quad(13)$$

Then $\mathcal{B}_{i,m}$ are disjoint length $T+k+1$ intervals except when $i=\lceil(b'_{max}-b'_{min}+1)/t\rceil$ and $m=\min\{4k+1,\lceil((b'_{max}-b'_{min}+1) \bmod t)/(T+2k+1)\rceil\}$ the length might be less. Let

$$\mathcal{IM}'=\{(i,m):|\mathcal{B}'_{i,m}|=T+k+1\}\quad(14)$$

be the set of (i,m) pairs for which $\mathcal{B}'_{i,m}$ has length $T+k+1$. Since $|\mathcal{I}^*_j|=|\mathcal{I}_j|-|\mathcal{I}_j\cap\boldsymbol{\delta}_w|$ for $w\in[1,d]$, we have that

$$b'_{max}-b'_{min}+1=|\mathcal{I}^*_j|\leq|\mathcal{I}_j|=b_{max}-b_{min}+1$$

It follows that $\mathcal{B}_{i,m}\neq\emptyset$ when $(i,m)\in\mathcal{IM}'$. For notation convenience, let $p_{i,m}$ and $q_{i,m}$ be the beginning and end points of interval $\mathcal{B}_{i,m}$, i.e., $\mathcal{B}_{i,m}=[p_{i,m},q_{i,m}]$ for $(i,m)\in\mathcal{IM}'$. Similarly, let $\mathcal{B}'_{i,m}=[p'_{i,m},q'_{i,m}]$ for $(i,m)\in\mathcal{IM}'$.

Algorithm 3: Determine the number of deletions

```
Input: The read matrix $\boldsymbol{D}\in\mathcal{D}_k(\mathbf{c},t)$ and the output
       intervals $\{\mathcal{I}^*_j\}_{j=1}^J$ of Algorithm 2
Output: $|\boldsymbol{\delta}_1\cap\mathcal{I}_j|$, $j\in[1,J]$
Step 1: Compute the intervals $\mathcal{B}'_{i,m}=[p'_{i,m},q'_{i,m}]$ using
  (13) and compute $\mathcal{IM}'$ using (14), $i,m\in\mathcal{IM}'$. Go to
  Step 2;
Step 2: For all $(i,m)\in\mathcal{IM}'$, find a unique
  integer $0\leq x_{i,m}\leq k$ such
  that $\boldsymbol{D}_{1,[p'_{i,m},q'_{i,m}-x_{i,m}]}=\boldsymbol{D}_{2,[p'_{i,m}+x_{i,m},q'_{i,m}]}$. If no or
  more than one such integers exist, let $x_{i,m}=0$. Go to
  Step 3;
Step 3: For all $m\in[1,4k+1]$, compute the
  sum $s_m=\sum_{i:(i,m)\in\mathcal{IJ}'}x_{i,m}$. Go to step 4;
Step 4: Output the majority among $\{s_m\}_{m=1}^{4k+1}$;
```

We now show that Algorithm 3 outputs $|\mathcal{I}_j\cap\boldsymbol{\delta}_1|$. It suffices to show that the term $s_{m_l}=|\mathcal{I}_j\cap\boldsymbol{\delta}_1|$ for $l\in[1,2k+1]$ in Algorithm 3. First, we show that the unique integer x_{i,m_l} satisfying $\boldsymbol{D}_{1,[p'_{i,m_l},q'_{i,m_l}-x_{i,m_l}]}=\boldsymbol{D}_{2,[p'_{i,m_l}+x_{i,m_l},q'_{i,m_l}]}$ exists for $l\in[1,2k+1]$ and i such that $(i,m_l)\in\mathcal{IM}'$. Moreover, the integer x_{i,m_l} equals $|\boldsymbol{\delta}_1\cap[p_{1,1},p_{i,m_l}-1]|-|\boldsymbol{\delta}_2\cap[p_{1,1},p_{i,m_l}-1]|$, the difference between the number of deletions in the first two heads that happen before the interval $\mathcal{B}_{i,m_l}$. Recall that m_l satisfies $\mathcal{U}_{m_l}\cap\boldsymbol{\delta}_w=\emptyset$ for $w\in\{1,2\}$ and that $\boldsymbol{D}_{w,p'_{1,1}}=\boldsymbol{D}_{w,b'_{min}}$ comes from $\mathbf{c}_{b_{min}}=\mathbf{c}_{p_{1,1}}$ after deletions for $w\in\{1,2\}$. Hence, the bit $\boldsymbol{D}_{w,p'_{i,m_l}}$ comes from $\mathbf{c}_{p_{i,m_l}+|\boldsymbol{\delta}_w\cap[p_{1,1},,p_{i,m_l}-1]|}$ after deletions for $w\in\{1,2\}$, by definitions of $p_{i,m}$ and $p'_{i,m}$. In addition, $\boldsymbol{D}_{w,[p'_{i,m_l},q'_{i,m_l}]}$ comes from $\mathbf{c}_{[p_{i,m_l}+|\boldsymbol{\delta}_w\cap[p_{1,1},,p_{i,m_l}-1]|,p_{i,m_l}+|\boldsymbol{\delta}_w\cap[p_{1,1},,p_{i,m_l}-1]|+T+k]}$. Let

$x=|\boldsymbol{\delta}_1\cap[p_{1,1},p_{i,m_l}-1]|-|\boldsymbol{\delta}_2\cap[p_{1,1},p_{i,m_l}-1]|$, we have that

$$\begin{aligned}&\boldsymbol{D}_{1,[p'_{i,m_l},q'_{i,m_l}-x]}\\ =&\mathbf{c}_{[p_{i,m_l}+|\boldsymbol{\delta}_1\cap[p_{1,1},p_{i,m_l}-1]|,p_{i,m_l}+|\boldsymbol{\delta}_1\cap[p_{1,1},p_{i,m_l}-1]|+T+k-x]}\\ =&\boldsymbol{D}_{2,[p'_{i,m_l}+x,q'_{i,m_l}]}.\end{aligned}\tag{15}$$

Therefore, the integer $x_{i,m_l}=x$ satisfies $\boldsymbol{D}_{1,[p'_{i,m_l},q'_{i,m_l}-x_{i,m_l}]}=\boldsymbol{D}_{2,[p'_{i,m_l}+x_{i,m},q'_{i,m_l}]}$. We show this x_{i,m_l} is unique. Suppose there exists another integer $y>x$ for which $\boldsymbol{D}_{1,[p'_{i,m_l},q'_{i,m_l}-y]}=\boldsymbol{D}_{2,[p'_{i,m_l}+y,q'_{i,m_l}]}$. For notation convenience, denote

$$P(i,m_l)\triangleq p_{i,m_l}+|\boldsymbol{\delta}_1\cap[p_{1,1},p_{i,m_l}-1]|.$$

Then we have that

$$\begin{aligned}&\boldsymbol{D}_{1,[p'_{i,m_l},q'_{i,m_l}-y]}\\ =&\boldsymbol{D}_{2,[p'_{i,m_l}+y,q'_{i,m_l}]}\\ \overset{(a)}{=}&\boldsymbol{D}_{1,[p'_{i,m_l}+y-x,q'_{i,m_l}-x]}\\ =&\mathbf{c}_{[P(i,m_l)+y-x,P(i,m_l)+T+k-x]},\end{aligned}$$

where (a) follows from Eq. (15). Since,

$$\boldsymbol{D}_{1,[p'_{i,m_l},q'_{i,m_l}-y]}=\mathbf{c}_{[P(i,m_l),P(i,m_l)+T+k-y]},$$

it follows that

$$\mathbf{c}_{[P(i,m_l)+y-x,P(i,m_l)+T+k-x]}=\mathbf{c}_{[P(i,m_l),P(i,m_l)+T+k-y]}.$$

It then follows that

$$\begin{aligned}&L(\mathbf{c}_{[P(i,m_l),P(i,m_l)+T+k-x]},y-x)\\ =&T+k-x+1\geq T+1,\end{aligned}$$

which is a contradiction to the fact that $L(\mathbf{c},\leq k)\leq T$. Similarly, such contradiction occurs when $y<x$. Hence such x_{i,m_l} is unique.

Next, we show that $s_{m_l}=|\boldsymbol{\delta}_1\cap\mathcal{I}_j|$ for $l\in[1,2k+1]$. Since $p_{i,m_l}-p_{i-1,m_l}=t$ for $i\in[2,\max_{(i,m_l)\in\mathcal{IM}'}i]$, we have that

$$\begin{aligned}&|\boldsymbol{\delta}_1\cap[p_{1,1},p_{i,m_l}-1]|\\ =&|\boldsymbol{\delta}_1\cap[p_{1,1},p_{1,m_l}-1]|+\sum_{w=1}^{i-1}|\boldsymbol{\delta}_1\cap[p_{w,m_l},p_{w+1,m_l}-1]|\\ \overset{(a)}{=}&|\boldsymbol{\delta}_2\cap[p_{2,1},p_{2,m_l}-1]|+\sum_{w=1}^{i-2}|\boldsymbol{\delta}_2\cap[p_{w+1,m_l},p_{w+2,m_l}-1]|\\ &+|\boldsymbol{\delta}_1\cap[p_{i-1,m_l},p_{i,m_l}-1]|\\ =&|\boldsymbol{\delta}_2\cap[p_{2,1},p_{i,m_l}-1]|+|\boldsymbol{\delta}_1\cap[p_{i-1,m_l},p_{i,m_l}-1]|\\ \overset{(b)}{=}&|\boldsymbol{\delta}_2\cap[p_{1,1},p_{i,m_l}-1]|+|\boldsymbol{\delta}_1\cap[p_{i-1,m_l},p_{i,m_l}-1]|,\end{aligned}$$

where (a) hold since $|\boldsymbol{\delta}_1\cap[p_{1,1},p_{1,m_l}-1]|=|\boldsymbol{\delta}_2\cap[p_{2,1},p_{2,m_l}-1]|$ and $|\boldsymbol{\delta}_1\cap[p_{w-1,m_l},p_{w,m_l}-1]|=|\boldsymbol{\delta}_2\cap[p_{w,m_l},p_{w+1,m_l}-1]|$ for $w\in[2,i-1]$. Equality (b) holds since $\mathcal{I}_j$ is deletion isolated and hence $\boldsymbol{\delta}_2\cap[p_{1,1},p_{2,1}-1]=\emptyset$. It then follows that $x_{i,m_l}=|\boldsymbol{\delta}_1\cap[p_{i-1,m_l},p_{i,m_l}-1]|$ $(p_{0,m_l}=p_{1,1})$ and that

$$s_{m_l}=|\boldsymbol{\delta}_1\cap[p_{1,1},p_{\max_{(i,m_l)\in\mathcal{IM}'}i,m_l}-1]|$$

Note that $\boldsymbol{\delta}_1\cap[p_{\max_{(i,m)\in\mathcal{IM}'}i,m_l},b_{max}]\subseteq\boldsymbol{\delta}_1\cap[b_{max}-t+1,b_{max}]$. Since $\boldsymbol{\delta}_1\cap[b_{max}-t+1,b_{max}]=\emptyset$ because $\mathcal{I}_j$ is deletion isolated, we have that $s_{m_l}=|\boldsymbol{\delta}_1\cap\mathcal{I}_j|$. Then the majority rule works.

V. Correcting $k\geq 2d$ Deletions

In this section we present the code for correcting $k\geq 2d$ deletions as well as a lower bound on the redundancy when $t=n^{o(1)}$. The code construction is similar to the one presented in Section III. We use Lemma 7 to identify the deletion indices within a set of disjoint intervals $\mathcal{I}_j$, each with length no more than B. Note that in order to apply Lemma 7, the sequence $\mathbf{c}\in\{0,1\}^n$ has to be transformed into a sequence $F(\mathbf{c})\in\{0,1\}^{n+k+1}$ (see Lemma 5) that satisfies $L(F(\mathbf{c}),\leq k)\leq T$. Then we use a concatenated code construction. Specifically, to protect a sequence $\mathbf{c}\in\{0,1\}^{n+k+1}$ from k deletions, we split $\mathbf{c}$ into blocks $\mathbf{a}_i$, $i\in[1,\lceil(n+k+1)/B\rceil]$ of length B as in Eq. (5). Then the function S defined in Eq. (6), which is a concatenation of the mappings H (see Lemma 3) of $\mathbf{a}_i$, $i\in[1,\lceil(n+k+1)/B\rceil]$, can be used to corret k deletions in $\mathbf{c}$ (see Lemma 8). Finally, a Reed-Solomon code is used to protect the sequence S. The encoding function is as follows

$$Enc_2(\mathbf{c})=(F(\mathbf{c}),R_2^{'}(\mathbf{c}),R_2^{''}(\mathbf{c}))\tag{16}$$

where

$$\begin{aligned}R_2^{'}(\mathbf{c})&=RS_{2\lfloor k/d\rfloor}(S(F(\mathbf{c}))),\\ R_2^{''}(\mathbf{c})&=Rep_{k+1}(H(R_2^{'}(\mathbf{c}))).\end{aligned}\tag{17}$$

The sequence $S(\cdot)$ is defined in (6), and $RS_{2\lfloor k/d\rfloor}$ is the systematic Reed-Solomon code given in Lemma 1, that corrects $2\lfloor k/d\rfloor$ erasure errors. The length of $R_2^{'}(\mathbf{c})$ is $N_1=2\lfloor k/d\rfloor\max\{\log(n+k+1),4k\log B+o(\log B)\}=2\lfloor k/d\rfloor\max\{\log n,4k\log t\}+o(\log n)$. The length of $R_2^{''}(\mathbf{c})$ is $N_2=4k(k+1)\log N_1+O(\log N_1)=o(\log n)$. The length of $Enc_2(\mathbf{c})$ is $N=n+k+1+N_1+N_2=n+2\lfloor k/d\rfloor\max\{\log n,4k\log t\}+o(\log n)$.

Theorem 4. *The set $\mathcal{C}_2=\{Enc_2(\mathbf{c}):\mathbf{c}\in\{0,1\}^n\}$ is a d-head k-deletion correcting code for $2d\leq k$, if the adjacent head distance t satisfies $t\geq\max\{(3k+\lceil\log n\rceil+2)[k(k-1)/2+1]+(7k-k^3)/6,(4k+1)(5k+\lceil\log n\rceil+3)\}$ for $i\in\{1,\ldots,d-1\}$. The code $\mathcal{C}_2$ can be constructed, encoded, and decoded in nt^{2k+1} time. The redundancy of $\mathcal{C}_2$ is $N-n=+2\lfloor k/d\rfloor\max\{\log n,4k\log t\}+o(\log n)$.*

Proof. The proof is essentially the same as the proof of Theorem 3. For any $\boldsymbol{D}\in\mathcal{D}_k(\mathbf{c},t)$, let $\mathbf{d}=\boldsymbol{D}_{1,[1,N-k]}$ be the first row of $\boldsymbol{D}$. The sequence $\mathbf{d}$ is a length $N-k$ subsequence of $Enc_2(\mathbf{c})$. Then it is possible to recover $H(R_2^{'}(\mathbf{c}))$ from the last N_2-k bits of d, which is a length N_2-k subsequence of $k+1$-fold repetition of $H(R_2^{'}(\mathbf{c}))$. Then, we can recover $R_2^{'}(\mathbf{c})$ from $\mathbf{d}_{[n+1,n+N_1-k]}$ by using $H(R_2^{'}(\mathbf{c}))$.

It suffices to show how to use $R'(\mathbf{c})$ to recover $F(\mathbf{c})$. According to Lemma 7, we can identify a set of $J\leq k$ deletion isolated intervals $\{\mathcal{I}_j\}_{j=1}^J$, each with length not greater than B, such that $\boldsymbol{\delta}_1\subseteq(\cup_{j=1}^J\mathcal{I}_j)$. All bits $F(\mathbf{c})_i$ with indices $i\in[1,n+k+1]\backslash(\cup_{j=1}^J\mathcal{I}_j)$ can be recovered using (4). Note that according to Lemma 6, the bits $\mathbf{c}_{\mathcal{I}_j}$ with $|\boldsymbol{\delta}_w \cap \mathcal{I}_j| \le d-1$ errors can be recovered, when $t \ge \max\{(3k + \lceil \log n \rceil + 2)[k(k-1)/2+1] + (7k-k^3)/6, (4k+1)(5k+\lceil \log n \rceil + 3)\}$. Moreover, each interval $\mathcal{I}_j$ with $|\boldsymbol{\delta}_w \cap \mathcal{I}_j| \ge d$ spans over at most two blocks $\mathbf{a}_i$. Note that $|\boldsymbol{\delta}_w \cap \mathcal{I}_j|$ is known by Lemma 7. Therefore, at most $2\lfloor k/d \rfloor$ blocks, the indices of which can be identified, contain at least d deletions. Hence the sequence $S(F(\mathbf{c}))$ can be recovered with at most $2\lfloor k/d \rfloor$ symbol erasures, with known erasure indices. With the help of the Reed-Solomon code redundancy $RS_{2\lfloor k/d \rfloor}(S(F(\mathbf{c})))$ (see Lemma 1), the sequence $S(F(\mathbf{c}))$ can be recovered. Then from Lemma 8 and Lemma 5 the sequence $F(\mathbf{c})$ and thus $\mathbf{c}$ can be recovered. The computation complexity of $Enc_2(\mathbf{c})$ has the same order as that of $Enc_1(\mathbf{c})$. It takes $O(nt^{2k+1})$ time to encode and decode $Enc_2(\mathbf{c})$. □

Now we present a lower bound on the redundancy for small head distances $t = n^{o(1)}$, which proves the last part of Theorem 1.

Theorem 5. *Let $\mathcal{C}$ be a d-head k-deletion code with length n. If the distance t satisfies $t = n^{o(1)}$ for $i \in [1, d-1]$, then we have that $|\mathcal{C}| \le 2^{\lfloor k/2d \rfloor \log n + o(\log n)}$.*

Proof. Sample the sequence $\mathbf{c}$ with period $2(d-1)t$,

$$\mathbf{c}' = (c_{1+(d-1)t}, c_{1+3(d-1)t}, \dots, c_{1+(2j+1)(d-1)t}, \dots, c_{1+(2\lfloor (n-1-(d-1)t)/2(d-1)t \rfloor - 1)(d-1)t})$$

We show that correcting k deletions in $\mathbf{c}$ is at least as hard as correcting $\lfloor k/d \rfloor$ erasures in $\mathbf{c}'$. It suffices to show that d deletions in heads $i \in [1,d]$ can erase the information of any bit in $\mathbf{c}'$. For $j \in [1, \lfloor (n-1-(d-1)t)/2(d-1)t \rfloor]$, let d deletions occur at positions

$$\{1 + (2j-1)(d-1)t - wt : w \in [0, d-1]\},$$

at head 1. Then the corresponding d deletion in head m occur with indices

$$\{1 + (2j-1)(d-1)t - wt + (m-1)t : w \in [0, d-1]\}$$

for $m \in [1,d]$. It follows that the bit $c_{1+(2j-1)(d-1)t}$ is deleted in all heads. Suppose a genie tells the indices and values of all the d deleted bits in each head except the value of the bit $c_{1+(2j-1)(d-1)t}$. Then this reduces to a erasure of the bit $c_{1+(2j-1)(d-1)t}$ in $\mathbf{c}'$. Note that in this way, k deletions in $\mathbf{c}$ can cause $\lfloor k/d \rfloor$ erasures in $\mathbf{c}'$. From the Hamming bound, the size $|\mathcal{C}|$ is upper bounded by

$$\begin{aligned} |\mathcal{C}| \le& 2^n / \left(\sum_{i=1}^{\lfloor k/2d \rfloor} \binom{\lfloor (n-1-(d-1)t)/2(d-1)t \rfloor}{i} \right) \\ =& 2^{n - \lfloor k/2d \rfloor (\log n - \log(2(d-1)t)) + o(\log n)} \\ =& 2^{n - \lfloor k/2d \rfloor \log n + o(\log n)}. \end{aligned}$$

□

According to Theorem 5, the redundancy of a d-head k-deletion code is lower bounded by $\lfloor k/2d \rfloor \log n + o(\log n)$.

VI. Correcting k Deletions and Insertions

In this section we show how to correct a combination of up to k deletions and insertions in the d-head racetrack memory. In this scenario, more challenges arise since there may not be "shifts" between different reads, as we observed in Lemma 9, after a combination of deletions and insertions. This makes detection and correction of errors more complicated. Moreover, Lemma 6 does not apply.

The encoding and decoding algorithms for this task can be regarded as a generalization of the algorithms for correcting k deletions. Similar to Section III and Section V, we notice that the error indices $(\boldsymbol{\delta}_i, \boldsymbol{\gamma}_i)$, $i \in [1,d]$ are contained in a set of disjoint edit isolated intervals (the definition of edit isolated intervals will be given later), each with bounded length. We can identify some of the intervals (see Algorithm 4) that contain edit isolated intervals. However, different from the cases in Section III and Section V, some of the edit isolated intervals cannot be detected and identified from the reads. Fortunately, the intervals that cannot be detected contain at least $2d$ errors in each read. In addition, the "shift" in bits outside the edit isolated intervals, caused by the errors in those edit isolated intervals, can be determined in a similar manner (see algorithm 5) to Algorithm 3. Therefore, the bits outside the edit isolated intervals can be recovered (see Algorithm 6). For identifiable edit isolated intervals, we provide a insertion/deletion counterpart (see Algorithm 7) of Lemma 6 (correcting deletion errors). Specifically, we will show how to correct less than d insertions/deletions in an identifiable edit isolated interval, which is more complicated than correct deletions only since there might be no bit shifts among reads. Then, we are left to correct the remaining identifiable $\lfloor k/d \rfloor$ intervals that contain at least d errors and undetactable intervals that contain at least $2d$ errors. For this part, we follow similar techniques in Section III and Section V and use Reed-Solomon codes to protect a sequence of deletion/insertion correcting mappings for all blocks, thereby achieving the $2\lfloor k/d \rfloor \max\{\log n, 4k \log t\} + o(\log n)$ redundancy. In the following, we provide the definition of edit isolated intervals.

Definition 2. *Let $\boldsymbol{\delta}_i = \{\delta_{i,1}, \dots, \delta_{i,r}\}$ and $\boldsymbol{\gamma}_i = \{\gamma_{i,1}, \dots, \gamma_{i,s}\}$ be the sets of deletion and insertion indices, respectively, in the ith head of a d-head racetrack memory, i.e. $\boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t$ and $\boldsymbol{\gamma}_{i+1} = \boldsymbol{\gamma}_i + t$, for $i \in [1, d-1]$. An interval $\mathcal{I}$ is* edit isolated *if*

$$\begin{aligned} \boldsymbol{\delta}_{i+1} \cap \mathcal{I} =& t + \boldsymbol{\delta}_i \cap \mathcal{I}, \text{ and} \\ \boldsymbol{\gamma}_{i+1} \cap \mathcal{I} =& t + \boldsymbol{\gamma}_i \cap \mathcal{I}. \end{aligned}$$

for $i \in [1, d-1]$.

We begin with the the algorithm for identifying a set of disjoint intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$, given the read matrix $E \in \mathcal{E}_k(\mathbf{c}, t)$, such that for each $j \in [1, J]$, there is an interval $[p_{1j}, p_{2j}]$ satisfying:

- **(A)** $[p_{1j}, p_{2j}] \subseteq [b_{1j}, b_{2j}]$
- **(B)** $\boldsymbol{E}_{w,i} = \boldsymbol{E}_{w',i}$ for any $w, w' \in [1,d]$ and $i \in ([b_{1j}, p_{1j} - 1] \cup [p_{2j}+1, b_{2j}])$
- **(C)** $\boldsymbol{E}_{[1,d],[p_{1j},p_{2j}]} \in \mathcal{E}_{k'}(\mathbf{c}_{\mathcal{I}_j})$ for some edit isolated interval $\mathcal{I}_j$ and $k' \ge 1$.

(D)
$$|[b_{1j}, b_{2j}]| \leq (2kdt + 2t + 1)(k + 1) + kdt + 2k \triangleq B_o \quad (18)$$
for $j \in [1, J]$.

(E) $\boldsymbol{E}_{w,i} = \boldsymbol{E}_{w',i}$ for any $w, w' \in [1, d]$ and $i \in [1, n + 1]\backslash(\cup_{j=1}^{J}[b_{1j}, b_{2j}])$.

Algorithm 4: Finding intervals $[b_{1j}, b_{2j}]$**,** $j \in [1, J]$

```
Input: The read matrix $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c}, t)$
Output: Intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$ satisfying
        properties (A), (B), (C), and (D)
Initialization: Set all integers $m \in [1, n']$ unmarked,
  where $n'$ is the number of columns in $\boldsymbol{E}$. Let $i = 1$.
  Find the largest positive integer $L$ such that the
  sequences $\boldsymbol{E}_{w,[i,i+L-1]} = \boldsymbol{E}_{w',[i,i+L-1]}$ for
  any $w, w' \in [1, d]$. If such $L$ exists and satisfies
  $L > kdt + t$, mark the integers $m \in [1, L - (kdt + t)]$
  and go to Step 1. Otherwise, go to Step 1;
Step 1: Find the largest positive integer $L$ such that the
  sequences $\boldsymbol{E}_{w,[i,i+L-1]} = \boldsymbol{E}_{w',[i,i+L-1]}$ for
  any $w, w' \in [1, d]$. Go to Step 2. If no such $L$ is found,
  set $L = 0$ and go to Step 2;
Step 2: If $L \geq 2(kdt + t) + 1$, mark the integers
  $m \in [i + kdt + t, \min\{i + L - 1, n'\} - (kdt + t)]$.
  Set $i = i + L + 1$ and go to Step 3. Else $i = i + 1$ and
  go to Step 3;
Step 3: If $i \leq n'$, go to Step 1. Else go to Step 4;
Step 4: Output all unmarked intervals;
```

The algorithm is presented in Algorithm 4 and is similar to the one in Section IV-A. However, different from the intervals $\mathcal{I}_j^*$, $j \in [1, J]$ generated in Section IV-A, which satisfy properties **(P1)** and **(P2)** in Section IV, here we do not necessarily have an edit isolated interval $\mathcal{I}'_j$ satisfying $\boldsymbol{E}_{[1,d],[b_{1j},b_{2j}]} \in \mathcal{E}_{k'}(\mathbf{c}_{\mathcal{I}'_j})$ for every $j \in [1, J]$. Also, the error indices $(\boldsymbol{\gamma}_w \cup \boldsymbol{\delta}_w)$, $w \in [1, d]$ may not be contained in the collection of intervals $\cup_{j=1}^{J}\mathcal{I}_j$.

In the following lemma, we show that the output intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$, of Algorithm 4 satisfy the properties **(A)**, **(B)**, **(C)**, **(D)**, and **(E)** above.

Lemma 10. *For a read matrix* $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c}) \in \{0, 1\}^{d\times n'}$*, Let* $[b_{1j}, b_{2j}]$*,* $j \in [1, J]$ *be the output intervals in the above procedure such that* $b_{11} < b_{12} < \ldots < b_{1J}$*. There exists a set of intervals* $[p_{1j}, p_{2j}]$*,* $j \in [1, J]$*, satisfying* ***(A)**, **(B)**, **(C)**, **(D)**, and **(E)** above.*

Proof. Note that for each interval $[b_{1j}, b_{2j}]$, we have $\boldsymbol{E}_{w,[b_{1j},b_{1j}+kdt+t-1]} = \boldsymbol{E}_{w',[b_{1j},b_{1j}+kdt+t-1]}$ and $\boldsymbol{E}_{w,[b_{2j}-kdt-t+1,b_{2j}]} = \boldsymbol{E}_{w',[b_{2j}-kdt-t+1,b_{2j}]}$ for any $w, w' \in [1, d]$, except for $j = 1$, $\boldsymbol{E}_{w,[b_{1j},b_{1j}+kdt+t-1]}$ may not be equal to $\boldsymbol{E}_{w',[b_{1j},b_{1j}+kdt+t-1]}$, in which case, we let that $p_{11} = 1$ and the following arguments hold. Consider the set of intervals $[b_{1j} + (i - 1)t, b_{1j} + it - 1]$ for $i \in [1, kd + 1]$. Note that an error occurs in at most d intervals, each in one of the d heads. Therefore, at most kd intervals contain errors. Then, there exists an interval $[b_{1j} + (i_1 - 1)t, b_{1j} + i_1t - 1]$ for some $i_1 \in [1, kd + 1]$ such that $[b_{1j} + (i_1 - 1)t, b_{1j} + i_1t - 1] \cap (\boldsymbol{\gamma}_w \cup \boldsymbol{\delta}_w) = \emptyset$ for $w \in [1, d]$. Similarly, there exists an interval $[b_{2j} - i_2t + 1, b_{2j} - (i_2 - 1)t]$ for some $i_2 \in [1, kd + 1]$, such that $[b_{2j} - i_2t + 1, b_{2j} - (i_2 - 1)t] \cap (\boldsymbol{\gamma}_w \cup \boldsymbol{\delta}_w) = \emptyset$ for $w \in [1, d]$. This implies that $[b_{1j} + i_1t - 1 - k, b_{2j} - i_2t + 1 + k]$ is an edit isolated interval. Let $\boldsymbol{E}_{[1,d],[p_{1j},p_{2j}]} \in \mathcal{E}_{k'_j}(\mathbf{c}_{[b_{1j}+i_1t-1-k,b_{2j}-i_2t+1+k]})$, where $k'_j = |[b_{1j}+i_1t-1-k, b_{2j}-i_2t+1+k] \cap \boldsymbol{\delta}_1| + |[b_{1j}+i_1t-1-k, b_{2j} - i_2t + 1 + k] \cap \boldsymbol{\gamma}_1|$, be the read matrix obtained from $\mathbf{c}_{[b_{1j}+i_1t-1-k,b_{2j}-i_2t+1+k]}$ after deletion errors with indices $\boldsymbol{\delta}_w \cap [b_{1j} + i_1t - 1 - k, b_{2j} - i_2t + 1 + k]$ and insertion errors with indices $\boldsymbol{\gamma}_w \cap [b_{1j} + i_1t - 1 - k, b_{2j} - i_2t + 1 + k]$, $w \in [1, d]$. Then we have that $p_{1j} \in [b_{1j} + t - 1 - 2k, b_{1j} + kdt + t - 1]$ and $p_{2j} \in [b_{2j} - kdt - t + 1, b_{2j} - t + 1 + 2k]$. Therefore, the intervals $[p_{1j}, p_{2j}]$, $j \in [1, J]$ satisfy **(A)**, **(B)**. To show that $[p_{1j}, p_{2j}]$, $j \in [1, J]$ satisfy **(C)**, we need to show $k'_j \geq 1$ for each j. Suppose on the contrary, $k'_j = 0$. Then since $\boldsymbol{E}_{[1,d],[p_{1j},p_{2j}]} \in \mathcal{E}_{k'_j}(\mathbf{c}_{[b_{1j}+i_1t-1-k,b_{2j}-i_2t+1+k]})$, we have that $\boldsymbol{E}_{w,[p_{1j},p_{2j}]} = \boldsymbol{E}_{w',[p_{1j},p_{2j}]}$ for any $w, w' \in [1, d]$. Then we have $\boldsymbol{E}_{w,[b_{1j},b_{2j}]} = \boldsymbol{E}_{w',[b_{1j},b_{2j}]}$ for any $w, w' \in [1, d]$, and $b_{1j} + kdt + t$ should have been marked, a contradiction to the fact that $[b_{1j}, b_{2j}]$ is an unmarked interval.

Next, we show that $|[b_{1j}, b_{2j}]| < (2kdt + 2t + 1)(k + 1) + kdt + 2k$. Note that an error that occurs with index i in the first head also occurs with index $i + (w - 1)t$ in the wth head. These indices are contained in an interval $[i, i + (d - 1)t]$ of length less than dt. The indices of k errors in d heads are contained in k intervals, each of length at most dt. If $|[b_{1j}, b_{2j}]| \geq (2kdt + 2t + 1)(k + 1) + kdt + 2k$, there exists a sub-interval $[b'_{1j}, b'_{2j}] \subseteq [b_{1j} + k, b_{2j} - k]$ with length at least $2kdt + 2t + 1$, that is disjoint with the k intervals that contain the indices of all errors in all heads. Therefore, $[b'_{1j}, b'_{2j}] \cap (\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) = \emptyset$ for $w \in [1, d]$. Since the interval $[b'_{1j}, b'_{2j}]$ has length more than t, the intervals $[1, b'_{1j} - 1]$ and $[b'_{2j} + 1, n + k + 1]$ are edit isolated, where $n + k + 1$ is the length of $\mathbf{c}$. Moreover, $\boldsymbol{E}_{w,i} = \boldsymbol{E}_{w',i}$ for any $w, w' \in [1, d]$ and $i \in [b'_{1j} - |\boldsymbol{\delta}_1 \cap [1, b'_{1j} - 1]| + |\boldsymbol{\gamma}_1 \cap [1, b'_{1j} - 1]|, b'_{2j} - |\boldsymbol{\delta}_1 \cap [1, b'_{1j} - 1]| + |\boldsymbol{\gamma}_1 \cap [1, b'_{1j} - 1]|]$. This implies that $i = b'_{1j} - |\boldsymbol{\delta}_1 \cap [1, b'_{1j} - 1]| + |\boldsymbol{\gamma}_1 \cap [1, b'_{1j} - 1] + kdt + t$ should be marked, contradicting to the fact that $[b'_{1j}, b'_{2j}]$ is unmarked, $j \in [1, J]$. Therefore, we proved **(D)**. Finally, for marked indices i, we have that $\boldsymbol{E}_{w,i} = \boldsymbol{E}_{w',i}$ for any $w, w' \in [1, d]$. Theorefore, we have **(E)**. □

In the remaining of this section, we first show how to determine the shifts caused by errors in the edit isolated intervals that can be detected. The algorithm for determining shifts is presented in Algorithm 5. This provides a way to correct most of the bits in $\mathbf{c}$, the algorithm for which is given in Algorithm 6. Then, we show how to correct $k < d$ deletions and insertions in total, and show that when $k \geq d$ and the the errors are not corrected, there is a constrait on the number of errors that occur. The algorithm for correcting $k < d$ deletions and insertions is summarized in Algorithm 7. Finally, we present our encoding and decoding algorithms for the general cases when $k \geq d$. The code is the same as the construction in Section V, but with a different decoding algorithm. Before dealing with the $k < d$ case, we present a proposition that is repeatedly used in this section.

Proposition 1. *Let $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c})$ be a read matrix for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}, \leq k) \leq T$. For any integers $i \in [1,n]$ and $w, w' \in [1,d]$ such that no error occurs in interval $[i-T-2k, i]$ in the wth and w'th head, i.e.,*

$$\begin{aligned}&(\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i-T-2k, i] = \emptyset, \text{ and}\\ &(\boldsymbol{\delta}_{w'} \cup \boldsymbol{\gamma}_{w'}) \cap [i-T-2k, i] = \emptyset,\end{aligned} \tag{19}$$

If

$$\boldsymbol{E}_{w,[i-T-2k,i-x]} = \boldsymbol{E}_{w',[i-T-2k+x,i]} \tag{20}$$

for some integer $x \in [0,k]$, then

$$\begin{aligned}&\Gamma(w,i)+x\\ =&|\boldsymbol{\gamma}_{w'} \cap [1, i-T-2k-1]| - |\boldsymbol{\delta}_{w'} \cap [1, i-T-2k-1]|\end{aligned} \tag{21}$$

Proof. Suppose on the contrary,

$$\begin{aligned}&|\boldsymbol{\gamma}_w \cap [1, i-T-2k-1]| - |\boldsymbol{\delta}_w \cap [1, i-T-2k-1]| + x'\\ =&|\boldsymbol{\gamma}_{w'} \cap [1, i-T-2k-1]| - |\boldsymbol{\delta}_{w'} \cap [1, i-T-2k-1]|\end{aligned} \tag{22}$$

for some $x' \neq x$. Denote

$$\Gamma(w,i) \triangleq |\boldsymbol{\gamma}_w \cap [1, i-T-2k-1]| - |\boldsymbol{\delta}_w \cap [1, i-T-2k-1]|$$

If $x' > x$, then we have that

$$\begin{aligned}&c_{[i-T-k+x'-x, i-k]}\\ \overset{(a)}{=}&\boldsymbol{E}_{w,[i-T-k+x'-x+\Gamma(w,i), i-k+\Gamma(w,i)]}\\ \overset{(b)}{=}&\boldsymbol{E}_{w',[i-T-k+x'+\Gamma(w,i), i-k+\Gamma(w,i)+x]}\\ \overset{(c)}{=}&\boldsymbol{E}_{w',[i-T-k+\Gamma(w',i), i-k+\Gamma(w',i)+x-x']}\\ \overset{(d)}{=}&c_{[i-T-k, i-k+x-x']},\end{aligned}$$

where (a) and (d) follows from (19) and the fact that $|\gamma_w| + |\delta_w| \leq k$ for $w \in [1,d]$, (b) follows from (20), and (c) follows from (22).

If $x' < x$, we have that

$$\begin{aligned}&c_{[i-T-k, i-k-x+x']}\\ =&\boldsymbol{E}_{w,[i-T-k+\Gamma(w,i), i-k-x+x'+\Gamma(w,i)]}\\ =&\boldsymbol{E}_{w',[i-T-k+\Gamma(w,i)+x, i-k+x'+\Gamma(w,i)]}\\ =&\boldsymbol{E}_{w',[i-T-k+\Gamma(w',i)+x-x', i-k+\Gamma(w',i)]}\\ =&c_{[i-T-k+x-x', i-k]},\end{aligned}$$

In both cases, we have that $L(\mathbf{c}, |x-x'|) \geq T+1$, contradicting to the fact that $L(\mathbf{c}, \leq k) \leq T$. Hence, $x' = x$ and the proof is done. □

A. Determine Bits Outside Edit Isolated Intervals

The following lemma shows that the bit shifts caused by errors in intervals $\mathcal{I}_j$, $j \in [1,J]$ can be determined.

Lemma 11. *Let $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c})$ be a read matrix for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}, \leq k) \leq T$. Let the head distance t satisfy $t > (4K+1)(T+4k+1)$. If there is an interval $[b_1, b_2]$, an interval $[p_1, p_2] \subseteq [b_1, b_2]$, and an edit isolated interval $\mathcal{I}$ satisfying $\boldsymbol{E}_{[1,d],[p_1,p_2]} \in \mathcal{E}_{k'}(\mathbf{c}_{\mathcal{I}})$ for some $0 < k' \leq d-1$, and $\boldsymbol{E}_{w,j} = \boldsymbol{E}_{w',j}$ for any $w, w' \in [1,d]$ and $j \in ([b_1, p_1-1] \cup [p_2+1, b_2])$, then the number of bit shifts caused by errors in interval $\mathcal{I}$, which is $|\boldsymbol{\gamma}_w \cap \mathcal{I}| - |\boldsymbol{\delta}_w \cap \mathcal{I}|$, can be decided from $\boldsymbol{E}_{[1,d],[b_1,b_2]}$, for $w \in [1,d]$. Moreover, if $\boldsymbol{E}_{w,[b_1,b_2]} = \boldsymbol{E}_{w',[b_1,b_2]}$ for any $w, w' \in [1,d]$, then $|\boldsymbol{\gamma}_w \cap \mathcal{I}| = |\boldsymbol{\delta}_w \cap \mathcal{I}|$ for any $w \in [1,d]$.*

Proof. Similar to what we did in Section IV-B. consider a set of intervals

$$\begin{aligned}\mathcal{B}_{i,m} = [&b_1 + (i-1)t + (m-1)(T+4k+1),\\ &b_1 + (i-1)t + m(T+4k+1) - 1],\\ &\text{for } m \in [1, 4k+1] \text{ and } i \in [0, \lceil \frac{b_2-b_1+1}{t} \rceil + 1]\\ &\text{satisfying } b_1 + (i-1)t + m(T+4k+1) - 1 \leq b_2.\end{aligned}$$

Note that the intervals $\mathcal{B}_{i,m}$ are disjoint when $t > (4k+1)(T+4k+1)$. For notation convenience, let

$$q_{i,m} \triangleq b_1 + (i-1)t + (m-1)(T+4k+1) \tag{23}$$

for $m \in [1, 4k+1]$ and $i \in [0, \lceil \frac{b_2-b_1+1}{t} \rceil + 1]$ satisfying $b_1 + (i-1)t + m(T+4k+1) - 1 \leq b_2$. Let

$$\mathcal{U}_m = \cup_{i: q_{i,m}-1 \leq b_2, i \in [1, \lceil \frac{b_2-b_1+1}{t} \rceil + 1]} \mathcal{B}_{i,m},$$

for $m \in [1, 4k+1]$. Since there are at most $2k$ errors in the first two heads, there are at least $(2k+1)$ choices of $m \in [1, 4k+1]$, $m_1, \ldots, m_{2k+1}$, such that $\mathcal{U}_{m_\ell} \cap (\boldsymbol{\delta}_1 \cup \boldsymbol{\gamma}_1 \cup \boldsymbol{\delta}_2 \cup \boldsymbol{\delta}_2) \cap \mathcal{I} = \emptyset$ for $\ell \in [1, 2k+1]$. For each $m \in [1, 4k+1]$ and integer $i \geq 1$ such that $q_{i,m} - 1 \leq b_2$, find the unique integer $x_{m,i} \in [0,k]$ such that

$$\begin{aligned}&\boldsymbol{E}_{1,[q_{i,m+1}+k, q_{i,m+2}-k-1-x_{m,i}]}\\ =&\boldsymbol{E}_{2,[q_{i,m+1}+k+x_{m,i}, q_{i,m+2}-k-1]}\end{aligned} \tag{24}$$

or $x_{m,i} \in [-k, -1]$ such that

$$\begin{aligned}&\boldsymbol{E}_{1,[q_{i,m+1}+k-x_{m,i}, q_{i,m+2}-k-1]}\\ =&\boldsymbol{E}_{2,[q_{i,m+1}+k, q_{i,m+2}-k-1+x_{\ell,i}]}\end{aligned} \tag{25}$$

If no such $x_{m,i}$ or more than one exist, let $x_{m,i} = k+1$. Denote

$$\Gamma'(w, i, m_l) \triangleq |\boldsymbol{\gamma}_w \cap [b_1, q_{i,m_\ell} - 1]| - |\boldsymbol{\delta}_w \cap [b_1, q_{i,m_\ell} - 1]|$$

. Since $[q_{i,m_\ell}, q_{i,m_\ell+1} - 1] \cap (\boldsymbol{\delta}_1 \cup \boldsymbol{\gamma}_1 \cup \boldsymbol{\delta}_2 \cup \boldsymbol{\delta}_2) \cap \mathcal{I} = \emptyset$, we have that

$$\begin{aligned}&\boldsymbol{E}_{1,[q_{i,m_\ell}+\Gamma'(1,i,m_l), q_{i,m_\ell+1}-1+\Gamma'(1,i,m_l)]}\\ =&\mathbf{c}_{[q_{i,m_\ell}, q_{i,m_\ell+1}-1]}\\ =&\boldsymbol{E}_{2,[q_{i,m_\ell}+\Gamma'(2,i,m_l), q_{i,m_\ell+1}-1+\Gamma'(2,i,m_l)]}\end{aligned}$$

which implies that the integer $x_{m_\ell,i} \in [-k,k]$ satisfying (24) and (25) can be found for $\ell \in [1, 2k+1]$. According to Proposition 1, such $x_{m_\ell,i}$ is unique. In the following, we show that

$$|\boldsymbol{\gamma}_w \cap \mathcal{I}| - |\boldsymbol{\delta}_w \cap \mathcal{I}| = \sum_{i: q_{i,m}-1 \leq b_2, i \in [1, \frac{b_2-b_1+1}{t}+1]} x_{m_\ell,i} \tag{26}$$

for $\ell \in [1, 2k+1]$.

For any fixed $\ell \in [1, 2k+1]$, let $i^* \in [0, \lceil \frac{b_2-b_1+1}{t} \rceil + 1]$ be the largest integer such that $(\boldsymbol{\gamma}_1 \cup \boldsymbol{\delta}_1) \cap \mathcal{I} \cap [1, q_{i^*,m+1} + k - 1] = \emptyset$. Note that $x_{m_\ell,i} = 0$ for $i \in [1, i^*]$, because $\mathcal{I}$ is edit isolated and $(\boldsymbol{\gamma}_w \cup \boldsymbol{\delta}_w) \cap \mathcal{I} \cap [1, q_{i^*,m+1} + k - 1] = \emptyset$ for $w \in [1, d]$. Hence, we have $|\boldsymbol{\gamma}_w \cap \mathcal{I} \cap [1, q_{i,m+1} - 1]| - |\boldsymbol{\delta}_w \cap \mathcal{I} \cap [1, q_{i,m+1} - 1| = 0 = \sum_{i=1}^{i^*} x_{m_\ell,i}$ for $w \in \{1, 2\}$. According to Proposition 1 and definition of $x_{m,i}$, we have that

$$\begin{aligned}
&x_{m_\ell,i}\\
=&|\boldsymbol{\gamma}_2 \cap [1, q_{i,m_\ell+1} + k - 1]| - |\boldsymbol{\delta}_2 \cap [1, q_{i,m_\ell+1} + k - 1]|\\
&- |\boldsymbol{\gamma}_1 \cap [1, q_{i,m_\ell+1} + k - 1]| + |\boldsymbol{\delta}_1 \cap [1, q_{i,m_\ell+1} + k - 1]|\\
\overset{(a)}{=}&|\boldsymbol{\delta}_1 \cap [q_{i-1,m_\ell+1} + k, q_{i,m_\ell+1} + k - 1]|\\
&- |\boldsymbol{\gamma}_1 \cap [q_{i-1,m_\ell+1} + k, q_{i,m_\ell+1} + k - 1]|
\end{aligned}$$

for $i \geq i^*+1$, where (a) follows since $|\boldsymbol{\gamma}_2 \cap [1, q_{i,m_\ell+1} + k - 1]| = |\boldsymbol{\gamma}_1 \cap [1, q_{i-1,m_\ell+1} + k - 1]|$ and $|\boldsymbol{\delta}_2 \cap [1, q_{i,m_\ell+1} + k - 1]| = |\boldsymbol{\delta}_1 \cap [1, q_{i-1,m_\ell+1} + k - 1]|$. Moreover $x_{m_\ell,i} = 0$ for $q_{i,m_\ell} \geq p_2 + 1$. Therefore,

$$\begin{aligned}
&\sum_{i:q_{i,m_\ell}-1 \leq b_2, i \in [1, \lceil \frac{b_2-b_1+1}{t} \rceil + 1]} x_{m_\ell,i}\\
=&\sum_{i:q_{i,m_\ell+1}-1 \leq p_2, i \geq i^*+1} (|\boldsymbol{\delta}_1 \cap [q_{i-1,m_\ell+1}+k, q_{i,m_\ell+1}+k-1]|\\
&- |\boldsymbol{\gamma}_1 \cap [q_{i-1,m_\ell+1} + k, q_{i,m_\ell+1} + k - 1]|)\\
=&|\boldsymbol{\delta}_1 \cap \mathcal{I}| - |\boldsymbol{\gamma}_1 \cap \mathcal{I}|,
\end{aligned}$$

where the last equality holds since $(\boldsymbol{\gamma}_1 \cup \boldsymbol{\delta}_1) \cap \mathcal{I} \cap [1, q_{i^*,m+1} + k - 1] = \emptyset$ and $\mathcal{I}$ is edit isolated. Therefore, we have (26) for $\ell \in [1, 2k+1]$. Find the majority of $\sum_{i:q_{i,m}-1 \leq b_2, i \in [1, \frac{b_2-b_1+1}{t}+1]} x_{m,i}$ for $m \in [1, 4k+1]$, we obtain the value $|\boldsymbol{\delta}_w \cap \mathcal{I}| - |\boldsymbol{\gamma}_w \cap \mathcal{I}|$ for $w \in [1, d]$.

Finally, since $x_{m,i} = 0$ for each pair of (m, i) when $\boldsymbol{E}_{w,[b_1,b_2]} = \boldsymbol{E}_{w',[b_1,b_2]}$ for any $w, w' \in [1, d]$, we have $|\boldsymbol{\gamma}_w \cap \mathcal{I}| = |\boldsymbol{\delta}_w \cap \mathcal{I}|$ for any $w \in [1, d]$. □

We summarize the algorithm described in the proof of Lemma 11 in Algorithm 5, which takes an output interval $[b_{1j}, b_{2j}]$ in Algorithm 4 for any $j \in [1, J]$ and the read matrix $\boldsymbol{E}$ as input, and find the bit shifts $|\boldsymbol{\gamma}_w \cap \mathcal{I}_j| - |\boldsymbol{\delta}_w \cap \mathcal{I}_j|$, for $w \in [1, d]$ and $j \in [1, J]$, where $\mathcal{I}_j$ is the edit isolated interval associated with $[b_{1j}, b_{2j}]$, $j \in [1, J]$.

The next lemma shows that we can recover most of the bits in $\mathbf{c}$. Before stating the lemma, we define the notion of a minimum edit isolated interval. An interval $\mathcal{I}$ is called a minimum edit isolated interval if there is no strict sub-interval $\mathcal{I}' \subsetneq \mathcal{I}$ of $\mathcal{I}$ that is edit isolated.

We note that the error indices in all heads are contained in a disjoint set of minimum isolated intervals.

Lemma 12. *Let $\{[b_{1j}, b_{2j}]\}_{j=1}^J$ be the set of output intervals in Algorithm 4 and let $\{\mathcal{I}_j\}_{j=1}^J$ be the corresponding edit isolated intervals. Let $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c})$ be a read matrix for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}, \leq k) \leq T$. For an index i not in any minimum edit isolated interval that is disjoint with $\mathcal{I}_j$, $j \in [1, J]$, if the column index of the bit $\boldsymbol{E}_{1,i-|[1:i-1]\cap\boldsymbol{\delta}_1|+|[1:i-1]\cap\boldsymbol{\gamma}_1|}$ coming from c_i in the first read is not contained in one of the output intervals $[b_{1j}, b_{2j}]$, the bit c_i can be correctly recovered given $\boldsymbol{E}$.*

```
Algorithm 5: Finding intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$
Input: The read matrix $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c}, t)$ and an output
       interval $[b_{1j}, b_{2j}]$ of Algorithm 4 for any $j \in [1, J]$
Output: $|\boldsymbol{\gamma}_w \cap \mathcal{I}_j| - |\boldsymbol{\delta}_w \cap \mathcal{I}_j|$, for $w \in [1, d]$ and
        $j \in [1, J]$
Step 1: Compute the number $q_{i,m}$ by using (23) and
  replacing $b_1$ by $b_{1j}$, for $m \in [1, 4k+1]$ and
  $i \in [0, \lceil \frac{b_{2j}-b_{1j}+1}{t} \rceil + 1]$ satisfying
  $b_{1j} + (i-1)t + m(T + 4k + 1) - 1 \leq b_{2j}$. Go to Step 2;
Step 2: For each $m \in [1, 4k+1]$ and $i \geq 1$ such that
  $q_{i,m} - 1 \leq b_{2j}$ find the unique integer $x_{m,i} \in [0, k]$
  satisfying (24) or $x_{m,i} \in [-k, -1]$ satisfying (25). If no
  such $x_{m,i}$ or more than one exist, let $x_{m,i} = k+1$. Go
  to Step 3;
Step 3: Output the majority of
  $\sum_{i:q_{i,m}-1 \leq b_{2j}, i \in [1, \frac{b_{2j}-b_{1j}+1}{t}+1]} x_{m,i}$ for $m \in [1, 4k+1]$;
```

Proof. Assume that $b_{11} < b_{12} < \ldots < b_{1J}$. For each output interval $[b_{1j}, b_{2j}]$, $j \in [1, J]$, let $q_j \in [b_{1j}, b_{2j}]$ be the largest integer such that there exist $w, w' \in [1, d]$ satisfying $\boldsymbol{E}_{w,q_j} \neq \boldsymbol{E}_{w',q_j}$. We show that $q_j \in [b_{1j} + k + 1, b_{2j} - k - 1]$ for $j \in [1, J]$, unless when $b_{11} = 1$ or $b_{2J} = n' \in [n+1, n+2k+1]$, we can assume that $b_{11} = -k-1$ and $b_{2J} = n+2k+2$, which does not affect the result. Note that for any index i such that there exist $w, w' \in [1, d]$ satisfying $\boldsymbol{E}_{w,i} \neq \boldsymbol{E}_{w',i}$, the indices $[i+1, i+kdt+t]$ are not marked and contained in some output interval. We have that $q_j \leq b_{2j} - k - 1$. Similarly, $q_j \geq b_{1j} + k + 1$ because the intervals $[q_j - kdt - t, q_j]$, $j \in [1, J]$ is not marked.

According to Lemma 10, each output interval $[b_{1j}, b_{2j}]$ is associated with an edit isolated interval $\mathcal{I}_j$, $j \in [1, J]$. Note that for any minimum edit isolated interval $[i_1, i_2]$ that is disjoint with $\mathcal{I}_j$ for $j \in [1, J]$, we have that

$$\boldsymbol{E}_{w,i} = \boldsymbol{E}_{w',i}$$

for any $w, w' \in [1, d]$ and $i \in [i_1, i_2]$. By Lemma 11, we have that $|[i_1, i_2] \cap \boldsymbol{\delta}_1| = |[i_1, i_2] \cap \boldsymbol{\gamma}_1|$, i.e., there is no bit shift caused by errors in interval $[i_1, i_2]$. In addition, the shift caused by errors in interval $\mathcal{I}_j$, $j \in [1, J]$, which is $|\mathcal{I}_j \cap \boldsymbol{\gamma}_1| - |\mathcal{I}_j \cap \boldsymbol{\delta}_1| = s_j$, can be determined. This implies that

$$\begin{aligned}
&|[1:i'-1] \cap \boldsymbol{\gamma}_1| - |[1:i'-1] \cap \boldsymbol{\delta}_1|\\
=&\sum_{j:b_{2j} < i' + |[1:i'-1]\cap\boldsymbol{\gamma}_1| - |[1:i'-1]\cap\boldsymbol{\delta}_1|} s_j\\
=&\sum_{j:q_j < i'} s_j\\
\overset{(a)}{=}&\sum_{j:q_j < i' + |[1:i'-1]\cap\boldsymbol{\gamma}_1| - |[1:i'-1]\cap\boldsymbol{\delta}_1|} s_j \qquad (27)
\end{aligned}$$

for any i' satisfying: (1) $i' - |[1:i'-1] \cap \boldsymbol{\delta}_1| + |[1:i'-1] \cap \boldsymbol{\gamma}_1|$ not in any output interval $[b_{1j}, b_{2j}]$, $j \in [1, J]$. (2) i' is not in any minimum edit isolated interval that is disjoint with $\mathcal{I}_j$, $j \in [1, J]$. The equality (a) holds because $q_j \in [b_{1j} + k +$

$1, b_{2j}-k-1]$, and $i' > q_j$ only when $b_{2j} < i' + |[1:i'-1] \cap \boldsymbol{\gamma}_1| + |[1:i'-1] \cap \boldsymbol{\delta}_1|$. For the same reason, $i' < q_j$ only when $b_{1j} > i' + |[1:i'-1] \cap \boldsymbol{\gamma}_1| + |[1:i'-1] \cap \boldsymbol{\delta}_1|$.

For any $\boldsymbol{E}_{1,i}$ such that i is not included in any output interval $[b_{1j}, b_{2j}]$, $j \in [1, J]$, let

$$c'_{i-\sum_{j:q_j<i} s_j} = \boldsymbol{E}_{1,i}. \tag{28}$$

be an estimate of the bit $c_{i-\sum_{j:q_j<i} s_j}$. Then, for any index i' such that $i' - |[1:i'-1] \cap \boldsymbol{\delta}_1| + |[1:i'-1] \cap \boldsymbol{\gamma}_1|$ is not included in any output interval and i' is not in any minimum edit isolated interval that is disjoint with $\mathcal{I}_j$, $j \in [1, J]$, we have that

$$\begin{aligned} &c_{i'} \\ =&\boldsymbol{E}_{1,i'-|[1:i'-1]\cap\boldsymbol{\delta}_1|+|[1:i'-1]\cap\boldsymbol{\gamma}_1|} \\ =&c'_{i'-|[1:i'-1]\cap\boldsymbol{\delta}_1|+|[1:i'-1]\cap\boldsymbol{\gamma}_1|-\sum_{j:q_j<i'-|[1:i'-1]\cap\boldsymbol{\delta}_1|+|[1:i'-1]\cap\boldsymbol{\gamma}_1|} s_j} \\ =&c'_{i'}, \end{aligned}$$

where the last equality follows from (27). Therefore, the proof is done. □

The algorithm for determining the bit c_i for i satisfying:

S1 i not in any minimum edit isolated interval that is disjoint with $\mathcal{I}_j$, $j \in [1, J]$.

S2 c_i not falling withing any interval $[b_{1,j}, b_{2j}]$, $j \in [1, J]$ in the first read after errors.

is given in Algorithm 6.

Algorithm 6: Recovering most of the bits c_i

```
Input: The read matrix $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c}, t)$ and the output
    intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$, of Algorithm 4
Output: $\mathbf{c}'$ such that $c'_i = c_i$ for $i$ satisfying S1 and S2
Step 1: Let $q_j \in [b_{1j}, b_{2j}]$ be the largest integer such that
    there exist $w, w' \in [1, d]$ satisfying $\boldsymbol{E}_{w,q_j} \neq \boldsymbol{E}_{w',q_j}$, for
    $j \in [1, J]$. Go to Step 2;
Step 2: For each $j \in [1, J]$, compute
    $s_j = |\mathcal{I}_j \cap \boldsymbol{\gamma}_1| - |\mathcal{I}_j \cap \boldsymbol{\delta}_1|$ using Algorithm 5. Go to
    Step 3;
Step 3: For any $i$ not included in any output interval
    $[b_{1j}, b_{2j}]$, $j \in [1, J]$, let $c'_{i-\sum_{j:q_j<i} s_j} = \boldsymbol{E}_{1,i}$ (see (28)).
    Output the sequence $\mathbf{c}'$;
```

B. Correcting $k < d$ Deletions and Insertions

The cases when $k < d$ are addressed in the following lemma, which proves the first part of Theorem 2, where $k < d$.

Lemma 13. *Let $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c})$ be a read matrix for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}, \leq k) \leq T$. Let the distance t satisfy $t > (\frac{k^2}{4} + 3k)(T + 3k + 1) + T + 5k + 1$. If there is an interval $[b_1, b_2]$, an interval $[p_1, p_2] \subseteq [b_1, b_2]$, and an edit isolated interval $\mathcal{I}$ satisfying $\boldsymbol{E}_{[1,d],[p_1,p_2]} \in \mathcal{E}_{k'}(\mathbf{c}_\mathcal{I})$ for some $k' \leq d-1$, and $\boldsymbol{E}_{w,j} = \boldsymbol{E}_{w',j}$ for any $w, w' \in [1, d]$ and $j \in ([b_1, p_1-1] \cup [p_2+1, b_2])$, then we can obtain a sequence $\mathbf{e} \in \{0,1\}^{p_1-b_1+b_2-p_2+|\mathcal{I}|}$ such that $\mathbf{e}_{[1,p_1-b_1]} = \boldsymbol{E}_{w,[b_1,p_1-1]}$ for $w \in [1, d]$, $\mathbf{e}_{[p_1-b_1+1,p_1-b_1+|\mathcal{I}|]} = \mathbf{c}_\mathcal{I}$, and $\mathbf{e}_{[p_1-b_1+|\mathcal{I}|+1,p_1-b_1+|\mathcal{I}|+b_2-p_2]} = \boldsymbol{E}_{w,[p_2+1,b_2]}$ for $w \in [1, d]$.*

Proof. Let i^* be the minimum index such that $i^* \geq b_1$ and there exist different $w, w' \in [1, d]$ satisfying $\boldsymbol{E}_{w,i^*} \neq \boldsymbol{E}_{w',i^*}$. Let $\boldsymbol{E}_{w^*,i^*}$ be the minority bit among $\{\boldsymbol{E}_{w,i^*}\}_{w=1}^d$, i.e., there are at most $\lfloor \frac{d}{2} \rfloor$ bits among $\{\boldsymbol{E}_{w,i^*}\}_{w=1}^d$ being equal to $\boldsymbol{E}_{w^*,i^*}$. We will first show that there are edit errors occur near index i^* in the w^*th head, unless when the numbers of 1-bits and 0-bits among $\{\boldsymbol{E}_{w,i^*}\}_{w=1}^d$ are equal, edit errors occur near index i^* in the first head. To this end, we begin with the following proposition.

Proposition 2. *Let $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c})$ be a read matrix for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}, \leq k) \leq T$. Let $i^* > 0$ be an integer such that $\boldsymbol{E}_{w,[i^*-T-2k-1,i^*-1]} = \boldsymbol{E}_{w',[i^*-T-2k-1,i^*-1]}$ for any $w', w \in [1, d]$. For any $w_1, w_2 \in [1, d]$ such that no error occurs in interval $[i^*-T-2k-1, i^*+k-1]$ in the w_1th and w_2th head, i.e.,*

$$\begin{aligned} &(\boldsymbol{\delta}_{w_1} \cup \boldsymbol{\gamma}_{w_1}) \cap [i^*-T-2k-1, i^*+k-1] = \emptyset, \text{ and} \\ &(\boldsymbol{\delta}_{w_2} \cup \boldsymbol{\gamma}_{w_2}) \cap [i^*-T-2k-1, i^*+k-1] = \emptyset, \end{aligned} \tag{29}$$

the bits $\boldsymbol{E}_{w_1,i^*}$ and $\boldsymbol{E}_{w_2,i^*}$ are equal.

Proof. According to Proposition 1, we have that

$$\begin{aligned} &|\boldsymbol{\gamma}_{w_1} \cap [1, i^*-T-2k-2]| - |\boldsymbol{\delta}_{w_1} \cap [1, i^*-T-2k-2]| \\ =&|\boldsymbol{\gamma}_{w_2} \cap [1, i^*-T-2k-2]| - |\boldsymbol{\delta}_{w_2} \cap [1, i^*-T-2k-2]|. \end{aligned} \tag{30}$$

Then,

$$\begin{aligned} \boldsymbol{E}_{w_1,i^*} &\overset{(a)}{=} \mathbf{c}_{i^*-|\boldsymbol{\gamma}_{w_1} \cap [1,i^*-T-2k-2]|+|\boldsymbol{\delta}_{w_1} \cap [1,i^*-T-2k-2]|} \\ &\overset{(b)}{=} \mathbf{c}_{i^*-|\boldsymbol{\gamma}_{w_2} \cap [1,i^*-T-2k-2]|+|\boldsymbol{\delta}_{\boldsymbol{w_2}} \cap [1,i^*-T-2k-2]|} \\ &\overset{(c)}{=} \boldsymbol{E}_{w_2,i^*} \end{aligned}$$

where (a) and (c) follow from (29) and the fact that $|\boldsymbol{\gamma}_w \cap [1, i^*-T-2k-2]| - |\boldsymbol{\delta}_w \cap [1, i^*-T-2k-2]| \leq k$. Equality (b) follows from (30). □

From Proposition 2, we can easily conclude that there is at least one error in interval $[i^*-T-2k-1, i^*+k-1]$ in one of the heads, i.e., $(\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i^*-T-2k-1, i^*+k-1] \neq \emptyset$ for some $w \in [1, d]$. Otherwise the bits $\boldsymbol{E}_{w,i^*}$ are equal for all $w \in [1, d]$, contradicting to the definition of i^*.

Next, we need the following proposition.

Proposition 3. *Let $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c})$ be a read matrix for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}, \leq k) \leq T$. Let $i^* > 0$ be an integer such that $\boldsymbol{E}_{w,[1,i^*-1]} = \boldsymbol{E}_{w',[1,i^*-1]}$ for any $w', w \in [1, d]$.If $T^* \geq T+2k+1$ and $t > (k+1)T^*$, then the number of heads where at least one error occurs in interval $[i^*-T^*, i^*+k-1]$ is at most $\lfloor \frac{k+1}{2} \rfloor$, i.e.,*

$$|\{w : (\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i^*-T^*, i^*+k-1] \neq \emptyset\}| \leq \lfloor \frac{k+1}{2} \rfloor$$

Moreover, when $|\{w : (\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i^*-T^*, i^*+k-1] \neq \emptyset\}| = \frac{k+1}{2}$, at least one error occurs in $[i^*-T^*, i^*]$ in the first head, i.e., $(\boldsymbol{\delta}_1 \cup \boldsymbol{\gamma}_1) \cap [i^*-T^*, i^*] \neq \emptyset$.

Proof. Let $\{w : w \in [2, d], (\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i^*-T^*, i^*+k-1] \neq \emptyset\} = \{w_1, w_2, \ldots, w_M\}$ be the set of heads (not including the first head) that contains at least one error in interval $[i^*-T^*, i^*+k-1]$. Let $w_1 > w_2 > \ldots > w_M$. We will show that there exist a set of integers $i_1, i_2, \ldots, i_M \in [0,k]$ such that $i_1 \geq i_2 \geq \ldots \geq i_M$ and

$$\begin{aligned}&|(\boldsymbol{\delta}_1 \cap [i^*-T^*-(w_\ell-1)t-(T^*+k)i_\ell,\\&\qquad i^*-T^*-(w_\ell-2)t-(T^*+k)i_\ell-1]|\\&+|\boldsymbol{\gamma}_1 \cap [i^*-T^*-(w_\ell-1)t-(T^*+k)i_\ell,\\&\qquad i^*-T^*-(w_\ell-2)t-(T^*+k)i_\ell-1]|\\&\geq 2 \qquad (31)\end{aligned}$$

for $\ell \in [1,M]$. Note that the intervals $[i^*-T^*-(w_\ell-1)t-(T^*+k)i_\ell, i^*-T^*-(w_\ell-2)t-(T^*+k)i_\ell-1]$ are disjoint for different $\ell \in [1,M]$ and are within the interval $[-T^*-(T^*+k)(k+1), i^*-T^*-1]$, since $t > (k+1)(T^*+1)$ for $j \in [1,d]$ and $i_\ell \leq k$ for $\ell \in [1,M]$. Then, the number of errors in the first head is at least $2|\{w : (\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i^*-T^*, i^*+k-1] \neq \emptyset, w \in [2,d]\}| + \mathbb{1}((\boldsymbol{\delta}_1 \cup \boldsymbol{\gamma}_1) \cap [i^*-T^*, i^*+k-1] \neq \emptyset)$, where $\mathbb{1}(A)$ is the indicator that equals 1 when A is true and equals 0 otherwise. Hence, we have that

$$\begin{aligned}&2|\{w : (\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i^*-T^*, i^*+k-1] \neq \emptyset, w \in [2,d]\}|\\&+\mathbb{1}((\boldsymbol{\delta}_1 \cup \boldsymbol{\gamma}_1) \cap [i^*-T^*, i^*+k-1] \neq \emptyset)\\&\leq k\end{aligned}$$

Then, it can be easily verified that the proposition follows.

Now we find the set of non-negative integers $i_1 \geq i_2 \geq \ldots \geq i_M$ satisfying (31). Let $i_0 = k$. Starting from $\ell = 1$ to $\ell = M$, find the largest non-negative integer i_ℓ such that $i_\ell \leq i_{\ell-1}$ and no errors occur in interval $[i^*-T^*-(T^*+k)(i_\ell+1), i^*-T^*-(T^*+k)i_\ell-1]$ in the w_ℓth or the $(w_\ell-1)$th heads, i.e.,

$$\begin{aligned}&(\boldsymbol{\gamma}_{w_\ell} \cup \boldsymbol{\delta}_{w_\ell} \cup \boldsymbol{\gamma}_{w_\ell-1} \cup \boldsymbol{\delta}_{w_\ell-1})\\&\cap [i^*-T^*-(T^*+k)(i_\ell+1), i^*-T^*-(T^*+k)i_\ell-1]\\&=\emptyset. \qquad (32)\end{aligned}$$

We show that such an $\ell \in [1,M]$ can be found as long as $t > (T^*+k)(k+2)$. Note that in the above procedure, for each integer $i \in [i_\ell+1, k]$, there is at least an edit error occurring in interval $[i^*-T^*-(T^*+k)(i+1), i^*-T^*-(T^*+k)i-1]$ in one of the heads w, which corresponds to an error that occurs in interval $[i^*-T^*-(T^*+k)(i+1)-(w-1)t, i^*-T^*-(T^*+k)i-1-(w-1)t]$ in the first head. In addition, the intervals $[i^*-T^*-(T^*+k)(i+1)-(w-1)t, i^*-T^*-(T^*+k)i-1-(w-1)t]$ are disjoint for different pairs (i,w), as long as $t \geq (T^*+k)(k+2)$. Since there are at most k errors in the first head and there are $k+1$ choices of i_ℓ, such an i_ℓ satisfying (32) can be found.

Since $\boldsymbol{E}_{w_\ell,i} = \boldsymbol{E}_{w_\ell-1,i}$ for $i \in [i^*-T^*-(T^*+k)(i_\ell+1), i^*-T^*-(T^*+k)i_\ell-1]$, by Proposition 1 we have that

$$\begin{aligned}&|\boldsymbol{\gamma}_{w_\ell-1} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1]|\\&-|\boldsymbol{\delta}_{w_\ell-1} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1]|\\=&|\boldsymbol{\gamma}_{w_\ell} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1]|\\&-|\boldsymbol{\delta}_{w_\ell} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1]|. \qquad (33)\end{aligned}$$

On the other hand, we have that

$$\begin{aligned}&|\boldsymbol{\gamma}_{w_\ell-1} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1-t]|\\&-|\boldsymbol{\delta}_{w_\ell-1} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1-t]|\\=&|\boldsymbol{\gamma}_{w_\ell} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1]|\\&-|\boldsymbol{\delta}_{w_\ell} \cap [1, i^*-T^*-(T^*+k)(i_\ell+1)-1]|. \qquad (34)\end{aligned}$$

Eq. (33) and Eq. (34) imply that

$$\begin{aligned}&|\boldsymbol{\gamma}_{w_\ell-1} \cap [i^*-T^*-(T^*+k)(i_\ell+1)-t,\\&\qquad i^*-T^*-(T^*+k)(i_\ell+1)-1]|\\=&|\boldsymbol{\delta}_{w_\ell-1} \cap [i^*-T^*-(T^*+k)(i_\ell+1)-t,\\&\qquad i^*-T^*-(T^*+k)(i_\ell+1)-1]| \qquad (35)\end{aligned}$$

Since $(\boldsymbol{\gamma}_{w_\ell} \cup \boldsymbol{\delta}_{w_\ell}) \cap [i^*-T^*, i^*+k-1] \neq \emptyset$ by definition of w_ℓ, we have that

$$\begin{aligned}&(\boldsymbol{\gamma}_{w_\ell-1} \cup \boldsymbol{\delta}_{w_\ell-1}) \cap [i^*-T^*-t, i^*+k-1-t]\\\subseteq&(\boldsymbol{\gamma}_{w_\ell-1} \cup \boldsymbol{\delta}_{w_\ell-1}) \cap [i^*-T^*-(T^*+k)(i_\ell+1)-t,\\&\qquad i^*-T^*-(T^*+k)(i_\ell+1)-1]\\\neq&\emptyset\end{aligned}$$

Together with(35), we have that

$$\begin{aligned}&|\boldsymbol{\gamma}_{w_\ell-1} \cap [i^*-T^*-(T^*+k)(i_\ell+1)-t,\\&\qquad i^*-T^*-(T^*+k)(i_\ell+1)-1]|\\+&|\boldsymbol{\delta}_{w_\ell-1} \cap [i^*-T^*-(T^*+k)(i_\ell+1)-t,\\&\qquad i^*-T^*-(T^*+k)(i_\ell+1)-1]|\\\geq&2,\end{aligned}$$

which implies (31) because $\boldsymbol{\gamma}_{w_\ell-1} = \boldsymbol{\gamma}_1 + (w_\ell-2)t$ and $\boldsymbol{\delta}_{w_\ell-1} = \boldsymbol{\delta}_1 + (w_\ell-2)t$. Hence, the proof is done. □

Recall that $w^* \in [1,d]$ is a head index such that $\boldsymbol{E}_{w^*,i^*}$ is a minority bit among $\{\boldsymbol{E}_{w,i^*}\}_{w=1}^d$, i.e., there are at most $\frac{d}{2}$ bits among $\{\boldsymbol{E}_{w,i^*}\}_{w=1}^d$ that is equal to $\boldsymbol{E}_{w^*,i^*}$. By Proposition 2 and Proposition 3, we conclude that when $k < d$, we have that $(\boldsymbol{\delta}_{w^*} \cup \boldsymbol{\gamma}_{w^*}) \cap [i^*-T-2k-1, i^*+k-1] \neq \emptyset$, if the number of bits among $\{\boldsymbol{E}_{w,i^*}\}_{w=1}^d$ being equal to $\boldsymbol{E}_{w^*,i^*}$ is is less than $d/2$. If $k < d$ and the number of bits among $\{\boldsymbol{E}_{w,i^*}\}_{w=1}^d$ being equal to $\boldsymbol{E}_{w^*,i^*}$ is is exactly $d/2$, we have that $(\boldsymbol{\delta}_1 \cup \boldsymbol{\gamma}_1) \cap [i^*-T-2k-1, i^*+k-1] \neq \emptyset$.

Now we have found a w^* with

$$(\boldsymbol{\delta}_{w^*} \cup \boldsymbol{\gamma}_{w^*}) \cap [i^*-T-2k-1, i^*+k-1] \neq \emptyset. \qquad (36)$$

In the remaining part of the proof, we show how to use knowledge of w^* to correct at least one error for each head, and reduce the d-head case to a $(d-1)$-head case. Then, the lemma follows by induction, since the case when $d = 1$ is obvious. Assume that $w^* \leq d-1$. The procedure when $w^* = d$ will be similar.

Note that $(\boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_w) \cap [i^*-T-2k-1+(w-w^*)t, i^*+k-1+(w-w^*)t] \neq \emptyset$ by (36). Consider the set of intervals

$$\begin{aligned}&[i^*+2k+(\ell-1)(T+3k+1)+(w-w^*)t,\\&i^*+2k-1+\ell(T+3k+1)+(w-w^*)t]\end{aligned}$$

for $\ell \in [1, \frac{k^2}{4}+3k]$ and $w \in [1,d]$. For notation convenience, denote

$$v_{w,\ell} \triangleq i^* + 2k + (\ell-1)(T+3k+1) + (w-w^*)t \quad (37)$$

for $\ell \in [1, \frac{k^2}{4}+3k]$ and $w \in [1,d]$. For each pair $\ell \in [1, \frac{k^2}{4}+3k]$ and $w \in [1, d-1]$, find a unique integer $x_{w,\ell} \in [0,k]$, such that

$$\boldsymbol{E}_{w,[v_{w,\ell}, v_{w,\ell+1}-1-x_{w,\ell}]} = \boldsymbol{E}_{w+1,[v_{w,\ell}+x_{w,\ell}, v_{w,\ell+1}-1]} \quad (38)$$

or $x_{w,\ell} \in [-k,-1]$ such that

$$\boldsymbol{E}_{w,[v_{w,\ell}-x_{w,\ell}, v_{w,\ell+1}-1]} = \boldsymbol{E}_{w+1,[v_{w,\ell}, v_{w,\ell}+x_{w,\ell+1}-1]} \quad (39)$$

If no such integer or more than one exist, let $x_{w,\ell} = k+1$.

Given $x_{w,\ell}$, $\ell \in [1, \frac{k^2}{4}+3k]$ and $w \in [1,d]$, define a binary vector $\mathbf{z} \in \{0,1\}^{\frac{k^2}{4}+3k}$ as follows:

$$z_\ell = \begin{cases} 1, \text{if there exists a } w \in [1,d-1] \text{ such that} \\ \quad x_{w,\ell} = k+1, \\ 1, \text{if there exists a } w \in [1,d-1] \text{ such that} \\ \quad x_{w,\ell} \neq x_{w,\ell-1} \text{ and } x_{w,\ell}, x_{w,\ell-1} \in [-k,k], \\ 0, \text{else.} \end{cases} \quad (40)$$

for $\ell \in \frac{k^2}{4}+3k$. In (40), it is assumed that $x_{w,0} = x_{w,1}$ for $w \in [1,d-1]$.

Let $y^* = |(\boldsymbol{\gamma}_{w^*} \cup \boldsymbol{\delta}_{w^*} \cup \boldsymbol{\gamma}_{w^*+1} \cup \boldsymbol{\delta}_{w^*+1}) \cap [v_{w^*,1}-k, v_{w^*,\frac{k^2}{4}+3k}+T+4k]|$ be the number of errors that occur in interval $[v_{w^*,1}-k, v_{w^*,\frac{k^2}{4}+3k}+T+4k]$ in the w^*th or (w^*+1)th head. Note that $y^* = |(\boldsymbol{\gamma}_w \cup \boldsymbol{\delta}_w \cup \boldsymbol{\gamma}_{w+1} \cup \boldsymbol{\delta}_{w+1}) \cap [v_{w,1}-k, v_{w,\frac{k^2}{4}+3k}+T+4k]|$ for $w \in [1,d]$. Moreover, $\boldsymbol{E}_{w,[v_{w,1}, v_{w,\frac{k^2}{4}+3k}+T+3k]}$ can be obtained by a subsequence of $\mathbf{c}_{[v_{w,1}-k, v_{w,\frac{k^2}{4}+3k}+T+4k]}$ after at most y^* deletions and insertions in interval $[v_{w,1}-k, v_{w,\frac{k^2}{4}+3k}+T+4k]$ in the wth head, $w \in [1,d-1]$.

We first show that $y^* \leq k-1$. Note that the $|(\boldsymbol{\gamma}_{w^*} \cup \boldsymbol{\delta}_{w^*}) \cap [i^*+k, n']|$ errors that occur after index i^*+k in the w^*th head, occur after index $i^*+k+t > v_{w^*,\frac{k^2}{4}+3k}+T+4k+1$ in the (w^*+1)th head. Moreover, the errors that occur in interval $[i^*-T-2k-1, i^*+k-1]$ in the w^*th head occur after $i^*-T-2k-1+t > v_{w^*,\frac{k^2}{4}+3k}+T+4k+1$ in the (w^*+1)th head, since $t > (\frac{k^2}{4}+3k)(T+3k+1)+T+5k+1$. Recall that $(\boldsymbol{\delta}_{w^*} \cup \boldsymbol{\gamma}_{w^*}) \cap [i^*-T-2k-1, i^*+k-1] \neq \emptyset$. Hence, there are at most $k - |(\boldsymbol{\gamma}_{w^*} \cup \boldsymbol{\delta}_{\boldsymbol{w}^*}) \cap [i^*+k, n']| - 1 + |(\boldsymbol{\gamma}_{w^*} \cup \boldsymbol{\delta}_{\boldsymbol{w}^*}) \cap [i^*+k, n']| = k-1$ errors that occur in interval $[i^*+k, v_{w^*,\frac{k^2}{4}+3k}+T+4k]$ in the w^*th or (w^*+1)th head.

Next, we show that there are at most $(2k-2)$ 1 entries in $\mathbf{z}$. Note that a single error in interval $[i^*+k, v_{w^*,\frac{k^2}{4}+3k}+T+4k]$ in the w^*th or (w^*+1)th head affects the value of at most a single entry $x_{w,\ell}$ and the entries $x_{w,\ell+1}, \ldots, x_{w,\frac{k^2}{4}+3k}$ increase or decrease by 1 for $w \in [1,d]$. This generates at most two 1 entries in $\mathbf{z}$. Hence there are at most $2y^* \leq 2k-2$ 1 entries in $\mathbf{z}$.

Let y be the number of 1 runs in $\mathbf{z}$. We show that there exists a 0-run $(z_{i+1}, \ldots, z_{i+k-y+2})$ of length $k-y+2$, for some $i \in [0, \frac{k^2}{4}+2k+y]$, which indicates that

$$\begin{aligned}&\boldsymbol{E}_{w,[v_{w,i+1}, v_{w,i+k-y+3}-x_{w,i+1}-1]} \\ =&\boldsymbol{E}_{w+1,[v_{w,i+1}+x_{w,i+1}, v_{w,i+k-y+3}-1]}\end{aligned} \quad (41)$$

if $x_{w,i+1} \in [0,k]$ or

$$\begin{aligned}&\boldsymbol{E}_{w,[v_{w,i+1}-x_{w,i+1}, v_{w,i+k-y+3}-1]} \\ =&\boldsymbol{E}_{w+1,[v_{w,i+1}, v_{w,i+k-y+3}+x_{w,i+1}-1]}\end{aligned} \quad (42)$$

if $x_{w,i+1} \in [-k,-1]$, for every $w \in [1,d-1]$.

Suppose on the contrary, each 0 run has length no more than $k-y+1$. Note that there are at most $y+1$ 0 runs with y 1 runs. Therefore, the length of $\mathbf{z}$ is upper bounded by

$$\begin{aligned}\frac{k^2}{4}+3k \leq& (y+1)(k-y+1)+2k-2 \\ =& -y^2+ky+3k-1 \\ \leq& \frac{k^2}{4}+3k-1,\end{aligned}$$

a contradiction.

We have proved the existence of a 0 run $(z_{i+1}, \ldots, z_{i+k-y+2})$, which implies (41) and (42). We now show that there are at most $k-y+1$ errors occur in interval $[v_{w,i+1}, v_{w,i+k-y+3}-1]$ in the w and/or $(w+1)$th head, for $w \in [1,d-1]$. As mentioned above, a single error in interval $[i^*+k, v_{w^*,\frac{k^2}{4}+3k}+T+4k]$ in the w^*th or (w^*+1)th head affects the value of at most a single entry $x_{w,\ell}$ and the entries $(x_{w,\ell+1}, \ldots, x_{w,\frac{k^2}{4}+3k})$ increase or decrease by 1 for $w \in [1,d]$. This generates at most a single 1 run in $\mathbf{z}$. In addition, errors in interval $[v_{w,i+1}, v_{w,i+k-y+3}-1]$ in the w and/or $(w+1)$th head generate at most two 1 runs that include z_i and $z_{i+k-y+3}$. Therefore, there are at least $y-2$ 1 runs in $\mathbf{z}$ that are generated by at least $y-2$ errors in $[i^*+k, [i^*+k, v_{w^*,\frac{k^2}{4}+3k}+T+4k]] \backslash [v_{w,i+1}, v_{w,i+k-y+3}-1]$. Hence, the number of errors in interval $[v_{w,i+1}, v_{w,i+k-y+3}-1]$ in the w and/or $(w+1)$th head is at most $y^*-y+2 \leq k-y+1$.

Therefore, there exists an integer $\ell \in [i+1, i+k-y+2]$ such that no errors occur in interval $[v_{w,\ell}, v_{w,\ell+1}-1]$ in the w and/or $(w+1)$th head, which implies that $\boldsymbol{E}_{w+1,[p_1, v_{w,\ell}-|\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+|\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+k]}$ is obtained from $\mathbf{c}_{\mathcal{I}\cap[1,v_{w,\ell}+k]}$, after deletion errors with indices $\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]$ and insertion errors with indices $\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]$. Moreover,

$$\begin{aligned}&\boldsymbol{E}_{w,[v_{w,\ell}+k+1-|\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+|\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|-x_{w,\ell}, p_2]} \\ \overset{(a)}{=}&\boldsymbol{E}_{w,[v_{w,\ell}+k+1-|\boldsymbol{\delta}_{w}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+|\boldsymbol{\gamma}_{w}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|, p_2]},\end{aligned}$$

where (a) follows from Proposition 1, can be obtained from $\mathbf{c}_{\mathcal{I}\cap[v_{w,\ell}+k+1, n+k+1]}$, after deletion errors with indices $\boldsymbol{\delta}_w \cap \mathcal{I} \cap [v_{w,\ell}, n+k+1]$ and insertion errors with indices $\boldsymbol{\gamma}_w \cap \mathcal{I} \cap [v_{w,\ell}, n+k+1]$. Therefore, by concatenating

$$\boldsymbol{E}_{w+1,[p_1, v_{w,\ell}-|\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+|\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+k]}$$

and

$$\boldsymbol{E}_{w,[v_{w,\ell}+k+1-|\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+|\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|-x_{w,\ell}, p_2]},$$

we have a sequence obtained from $\mathbf{c}_{\mathcal{I}}$ by deletion errors with indices $(\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1])\cup(\boldsymbol{\delta}_w\cap\mathcal{I}\cap[v_{w,\ell},n+k+1])$ and insertion errors with indices $(\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1])\cup(\boldsymbol{\gamma}_w\cap\mathcal{I}\cap[v_{w,\ell},n+k+1])$, $w\in[1,d-1]$ in $\mathbf{c}_{\mathcal{I}}$. Note that there are at most $|\boldsymbol{\delta}_w\cap\mathcal{I}|+|\boldsymbol{\gamma}_w\cap\mathcal{I}|-1$ errors in total in the concatenation, since

$$|\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|=|\boldsymbol{\delta}_w\cap\mathcal{I}\cap[1,v_{w,\ell}-1-t]|,$$

and

$$|\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|=|\boldsymbol{\gamma}_w\cap\mathcal{I}\cap[1,v_{w,\ell}-1-t]|,$$

and the errors occur in $[(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap[v_{w,1}-T-4k-1,v_{w,1}-k-1]|]\neq\emptyset$ (see (36)) are not included in the concatenation. Finally, since $(z_{i+1},\dots,z_{i+k-y+2})$ is a 0 run, we have that $x_{w,i+1}=x_{w,i+2}=\dots=x_{w,i+k-y+2}$ for $w\in[1,d-1]$. Hence, concatenating $\boldsymbol{E}_{w+1,[p_1,v_{w,i+1}+k]}$ and $\boldsymbol{E}_{w,[v_{w,i+1}+k+1-x_{w,i+1},p_2]}$ results in the same sequence as concatenating $\boldsymbol{E}_{w+1,[p_1,v_{w,\ell}-|\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+|\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+k]}$ and $\boldsymbol{E}_{w,[v_{w,\ell}+k+1-|\boldsymbol{\delta}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|+|\boldsymbol{\gamma}_{w+1}\cap\mathcal{I}\cap[1,v_{w,\ell}-1]|-x_{w,\ell},p_2]}$, $w\in[1,d-1]$. Note that p_1,p_2 and $x_{w,\ell}$ are not known by the algorithm. We concatenate $\boldsymbol{E}_{w+1,[b_1,v_{w,i+1}+k]}$ and $\boldsymbol{E}_{w,[v_{w,i+1}+k+1-x_{w,i+1},b_2]}$ for each $w\in[1,d-1]$. Let the $d-1$ concatenated sequences be represented by a read matrix $\boldsymbol{E}'\in\{0,1\}^{(d-1)\times m'}$. Then from the above arguments, we have that $\boldsymbol{E}'_{[1,d-1],[p_1-b_1+1,m'-(b_2-p_2)]}\in\mathcal{E}_{k''}(\mathbf{c}_{\mathcal{I}})$ for some $k''\leq k'-1$. In addition, we have that $\boldsymbol{E}'_{w,[1,p_1-b_1]}=\boldsymbol{E}_{w,[b_1,p_1-1]}$ and $\boldsymbol{E}'_{w,[m'-(b_2-p_2)+1,m']}=\boldsymbol{E}_{w,[p_2+1,b_2]}$ for any $w\in[1,d-1]$.

For cases when $w^*=d$, the proof is similar, where instead of looking at intervals $[i^*+2k+(\ell-1)(T+3k+1)+(w-w^*)t,i^*+2k-1+\ell(T+3k+1)+(w-w^*)t]$ and defining $x_{w,\ell}$, $w\in[1,d-1]$, $\ell\in[1,\frac{k^2}{4}+3k]$ on these intervals, we define $x_{w,\ell}$ on intervals $[i^*-T-3k-\ell(T+3k+1)+(w-w^*)t,i^*-T-3k-1-(\ell-1)(T+3k+1)+(w-w^*)t]$ for $\ell\in[1,\frac{k^2}{4}+3k]$ and $w\in[1,d-1]$. Then we define z_ℓ based on $x_{w,\ell}$ and find a 0 run $(z_{i+1},\dots,z_{i+k-y+2})$ of length $k-y+2$ in $\mathbf{z}$, where y is the number of 1 runs in $\mathbf{z}$, and concatenate

$$\boldsymbol{E}_{w+1,[p_1,i^*-T-3k-(i+1)(T+3k+1)+(w-w^*)t+k]}$$

and

$$\boldsymbol{E}_{w,[i^*-T-3k-(i+1)(T+3k+1)+(w-w^*)t+k+1-x_{w,i+1},p_2]}$$

for $w\in[1,d-1]$.

We have shown how to correct at least one error using d reads. To correct all errors, we repeat the same procedure iteratively. In each iteration, we get a read matrix with one less read. The algorithm stops when we get a read matrix where all rows are equal. Let d^* be the number of rows of the read matrix after the algorithm stops. Let $\mathbf{e}$ be the first row of this matrix.

We complete the proof of Lemma 13 with the following proposition, which claims that either the errors contained in the isolated interval $\mathcal{I}$ can be corrected in $\mathbf{e}$ and the number of errors in $\mathcal{I}$ is at least $d-d^*$, or the errors contained in $\mathcal{I}$ cannot be corrected in $\mathbf{e}$ and the number of errors in $\mathcal{I}$ is at least $d+d^*$. In particular, for errors that are not detectable, i.e., when all the rows in $\boldsymbol{E}$ are equal, the number of errors contained in $\boldsymbol{E}$ is either 0 or at least $2d$. The proposition will be used to prove the correctness of the encoding/decoding algorithms in Section VI-C.

Proposition 4. *Let the length of* $\mathbf{e}$ *be* m. *Then,* $\mathbf{e}_{[1,p_1-b_1]}=\boldsymbol{E}_{w,[b_1,p_1-1]}$ *and* $\mathbf{e}_{[m-(b_2-p_2)+1,m]}=\boldsymbol{E}_{w,[p_2+1,b_2]}$ *for* $w\in[1,d]$. *In addition, the number of errors* $|(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap\mathcal{I}|\geq d-d^*$. *If* $\mathbf{e}_{[p_1-b_1+1,m-(b_2-p_2)]}\neq\mathbf{c}_{\mathcal{I}}$, *then the number of errors* $|(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap\mathcal{I}|\geq d+d^*$ *for* $d^*\geq2$. *When* $d^*=1$, *either* $|(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap\mathcal{I}|\geq d+d^*$ *or it can be determined that* $|(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap\mathcal{I}|\geq d$. *In particular, if the number of errors* $|(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap\mathcal{I}|\leq d-1$, *then* $\mathbf{e}_{[p_1-b_1+1,m-(b_2-p_2)]}=\mathbf{c}_{\mathcal{I}}$.

Proof. Let $\boldsymbol{E}^i$ be the read matrix obtained after the ith iteration, $i\in[1,d-d^*]$. Note that the number of rows in $\boldsymbol{E}^i$ is $d-i$. Let the number of columns in $\boldsymbol{E}^i$ be m_i.

Note that $\boldsymbol{E}_{w,[b_1,p_1-1]}$ and $\boldsymbol{E}_{w,[p_2+1,b_2]}$ are equal for $w\in[1,d]$. The concatenation in the algorithm keeps the first p_1-b_1 bits and the last b_2-p_2 bits in each row. Therefore, $\boldsymbol{E}^i_{w,[1,p_1-b_1]}=\boldsymbol{E}^i_{1,[b_1,p_1-1]}$ and $\boldsymbol{E}^i_{w,[m_i-(b_2-p_2)+1,m_i]}=\boldsymbol{E}_{1,[p_2+1,b_2]}$ for $w\in[1,d]$ and $i\in[1,d-d^*]$, which implies that $\boldsymbol{s}^j_{[1,p_1-b_1]}=\boldsymbol{E}^i_{w,[b_1,p_1-1]}$ and $\boldsymbol{s}^j_{[m-(b_2-p_2)+1,m]}=\boldsymbol{E}_{w,[p_2+1,b_2]}$ for $w\in[1,d]$.

Furthermore, we proved that $\boldsymbol{E}^1_{[1,d-1],[p_1-b_1+1,m_1-(b_2-p_2)]}\in\mathcal{E}_{k_1}(\mathbf{c}_{\mathcal{I}})$ for some $k_1\leq k'-1$. By induction, it can be proved that $\boldsymbol{E}^i_{[1,d-i],[p_1-b_1+1,m_i-(b_2-p_2)]}\in\mathcal{E}_{k_i}(\mathbf{c}_{\mathcal{I}})$ for some non-negative $k_i\leq k_{i-1}-1$ for $i\in[2,d-d^*]$. Therefore, we have that $0\leq k_{d-d^*}\leq k'-(d-d^*)$ and thus, $k'\geq d-d^*$.

If $\mathbf{e}_{[p_1-b_1+1,m-(b_2-p_2)]}=\boldsymbol{E}^{d-d^*}_{w,[1,m_{d-d^*}]}\neq\mathbf{c}_{\mathcal{I}}$ for $w\in[1,d^*]$, we let $\mathcal{I}=[i_1,i_2]$. When $d^*=1$, according to Lemma 12, the number of errors occur in $\mathcal{I}$ can be determined. Therefore, if the parity of the number of errors occur in $\mathcal{I}$ is different from the parity of $d+1$, we determine that the number of errors in $\mathcal{I}$ is at least d. Otherwise, the number of errors in $\mathcal{I}$ is at least $d+1$.

When $d^*\geq2$, let i' be the minimum index such that $i'\geq p_1-b_1+1$ satisfying $\boldsymbol{E}^{d-d^*}_{w,i'}\neq\mathbf{c}_{i_1+i'-(p_1-b_1)-1}$, i.e., $\boldsymbol{E}^{d-d^*}_{w,[p_1-b_1+1,i'-1]}=\mathbf{c}_{[i_1,i_1+i'-(p_1-b_1)-2]}$ and $\boldsymbol{E}^{d-d^*}_{w,i'}\neq\mathbf{c}_{i_1+i'-(p_1-b_1)-1}$. We show that $(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap[i'-T-2k'-1,i'+k'-1]\neq\emptyset$ for $w\in[1,d^*]$. Otherwise, there exists a $w^*\in[1,d^*]$, such that $(\boldsymbol{\delta}_{w^*}\cup\boldsymbol{\gamma}_{w^*})\cap[i'-T-2k'-1,i'+k'-1]=\emptyset$. Assume now that there is a virtual read $\boldsymbol{E}^{d-d^*}_{d^*+1,[1,m_{d-d^*}]}$. Assume that the distance between the d^*th head and the d^*+1th head is far enough so that the first error occurs after index i' in the read[5] $\boldsymbol{E}^{d-d^*}_{d^*+1,[1,m_{d-d^*}]}$. Therefore, $\boldsymbol{E}^{d-d^*}_{d^*+1,[p_1-b_1+1,i'-1]}=\mathbf{c}_{[i_1,i_1+i'-(p_1-b_1)-2]}=\boldsymbol{E}^{d-d^*}_{w,[p_1-b_1+1,i'-1]}$ for $w\in[1,d^*]$ and $\boldsymbol{E}_{d+1,i'}=\mathbf{c}_{i_1+i'-(p_1-b_1)-1}$. Applying Proposition 2 by considering a two-row matrix where the first row is $\boldsymbol{E}^{d-d^*}_{w^*,[1,m_{d-d^*}]}$ and the second row is $\boldsymbol{E}^{d-d^*}_{d^*+1,[1,m_{d-d^*}]}$, we have that $\boldsymbol{E}_{w^*,i'}=\boldsymbol{E}_{d^*+1,i'}=\mathbf{c}_{i_1+i^*-(p_1-b_1)-1}$, contradicting to the definition of i'. Hence, we have that $(\boldsymbol{\delta}_w\cup\boldsymbol{\gamma}_w)\cap[i'-T-2k'-1,i'+k'-1]\neq\emptyset$ for $w\in[1,d^*]$.

[5]This might require extending the length of the heads to larger than m_{d-d^*} when an error occurs near index m_{d-d^*} in the d^*th read in $\boldsymbol{E}^{d-d^*}$.

According to Proposition 3, we have that $k_{d-d^*} \geq 2d^* - 1$. Furthermore, by Lemma 11, we have that k_{d-d^*} is an even number and thus $k_{d-d^*} \geq 2d^*$. Therefore, we have that $k' \geq k_{d-d^*} + d - d^* \geq d + d^*$, when $\mathbf{e}_{[p_1-b_1+1, m-(b_2-p_2)]} \neq \mathbf{c}_{\mathcal{I}}$. □

The algorithm described in Lemma 13 for correcting errors in bits $\mathbf{c}_{\mathcal{I}_j}$, where $\mathcal{I}_j$ is associated with interval $[b_{1j}, b_{2j}]$, $j \in [1, J]$, is summarized in Algorithm 7.

Algorithm 7: Recovering most of the bits c_i

```
Input: The read matrix $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c}, t)$ and an output
       interval $[b_{1j}, b_{2j}]$ of Algorithm 4 for any $j \in [1, J]$
Output: Sequences $\mathbf{e}_j$ obtained from $\boldsymbol{E}_{1,[b_{1j},b_{2j}]}$ by
        correcting errors in interval $\mathcal{I}_j$ (the edit isolated
        interval associated with $[b_{1j}, b_{2j}]$) in the first
        head, when $|\boldsymbol{\delta}_1 \cap \mathcal{I}_j| + |\boldsymbol{\gamma}_1 \cap \mathcal{I}_j| < d$.
Initialization: Let $\boldsymbol{E}' = \boldsymbol{E}_{[1,d],[b_{1j},b_{2j}]}$;
Step 1: Let $i_j^*$ be the minimum positive index such that
  and there exist $w, w' \in [1, d]$ satisfying $\boldsymbol{E}'_{w,i_j^*} \neq \boldsymbol{E}'_{w',i_j^*}$,
  for some $j \in [1, J]$. Let $w_j^*$ be an index such that
  $\boldsymbol{E}'_{w_j^*,i_j^*}$ is the minority bit among $\{\boldsymbol{E}'_{w,i_j^*}\}_{w=1}^{d}$. If no
  such $i_j^*$ exists, go to step 6. Otherwise, go to Step 2;
Step 2: For each $\ell \in [1, \frac{k^2}{4} + 3k]$ and $w \in [1, d-1]$, find
  a unique integer $x_{w,\ell} \in [0, k]$ satisfying (38), or
  $x_{w,\ell} \in [-k, -1]$ satisfying (39), where the index $v_{w,\ell}$ in
  (38) and (39) is given by (37) when $w_j^* < d$. When
  $w_j^* = d$, the indices $v_{w,\ell}$ and $v_{w,\ell+1}$ are replaced by
  $i_j^* - T - 3k - \ell(T + 3k + 1) + (w - w^*)t$ and
  $i_j^* - T - 3k - 1 - (\ell - 1)(T + 3k + 1) + (w - w^*)t$,
  respectively. If no such integer or more than one exist,
  let $x_{w,\ell} = k + 1$. Go to Step 3;
Step 3: Let the vector $\mathbf{z} \in \{0,1\}^{\frac{k^2}{4}+3k}$ be defined as in
  (40). Find an index $i$ such that $(z_{i+1}, \ldots, z_{i+k-y+2})$ is
  a 0 run, where $y$ is the number of 1 runs in $\mathbf{z}$. Go to
  Step 4;
Step 4: Let $\boldsymbol{E}''$ be a new matrix. For each
  $w \in [1, d-1]$, let the $w$th row of $\boldsymbol{E}''$ be given by
  concatenating $\boldsymbol{E}'_{w+1,[1,v_{w,i+1}+k]}$ and
  $\boldsymbol{E}'_{w,[v_{w,i+1}+k+1,-x_{w,i+1}:\alpha_w]}$ where $\alpha_w$ is the length of
  the $w$th row of $\boldsymbol{E}'$, when $w_j^* < d$. When $w_j^* = d$, the
  $w$th row of $\boldsymbol{E}''$ is given by concatenating
  $\boldsymbol{E}'_{w+1,[1,i_j^*-T-3k-(i+1)(T+3k+1)+(w-w^*)t+k]}$ and
  $\boldsymbol{E}'_{w,[i_j^*-T-3k-(i+1)(T+3k+1)+(w-w^*)t+k+1-x_{w,i+1}:\alpha_w]}$.
  Go to Step 5;
Step 5: $\boldsymbol{E}'' = \boldsymbol{E}'$ and go to Step 1;
Step 6: Output the first row of $\boldsymbol{E}'$;
```

□

C. Encoding/Decoding Algorithms

We are now ready to present the encoding and decoding algorithms. We first deal with cases when $k \geq 2d$. Given any input sequence $\mathbf{c} \in \{0,1\}^n$, the encoding is similar to the one in Section V, stated in (16) and (17) and is given by

$$Enc_2(\mathbf{c}) = (F(\mathbf{c}), R_2^{'}(\mathbf{c}), R_2^{''}(\mathbf{c})), \tag{43}$$

where

$$\begin{aligned} R_2^{'}(\mathbf{c}) &= RS_{2\lfloor k/d \rfloor}(S(F(\mathbf{c}))), \\ R_2^{''}(\mathbf{c}) &= Rep_{k+1}(H(R_2^{'}(\mathbf{c}))). \end{aligned} \tag{44}$$

The difference here is in the definition of the function $S(F(\mathbf{c}))$, instead of splitting $F(\mathbf{c})$ into blocks of length B, as in (6), we split $F(\mathbf{c})$ into blocks of length $B' = B_o + k$, where B_o is the upper bound on the length of the output intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$, defined in (18), i.e.,

$$\begin{aligned} F(\mathbf{c}) =& (\mathbf{a}_1^{'}, \ldots, \mathbf{a}^{'}_{\lceil \frac{n+k+1}{B'} \rceil}), \text{ and} \\ S(F(\mathbf{c})) =& (H(\mathbf{a}_1^{'}), \ldots, H(\mathbf{a}^{'}_{\lceil \frac{n+k+1}{B'} \rceil})) \end{aligned} \tag{45}$$

It can be verified that the code has asymptotically the same redundancy $2\lfloor k/d \rfloor \max\{\log(n + k + 1), 4k \log B' + o(\log B')\} = 2\lfloor k/d \rfloor \max\{\log n, 4k \log t\} + o(\log n)$ as in deletion only cases. In the following, we show that the codeword $Enc_2(\mathbf{c})$ can be correctly decoded. We first show the following proposition.

Proposition 5. *For any $\mathbf{c}' \neq \mathbf{c}$ such that $\mathbf{c}'$ and $\mathbf{c}$ share the same redundancy $R_2^{'}(\mathbf{c}) = R_2^{'}(\mathbf{c}')$, we have that $\boldsymbol{E} \notin \mathcal{E}_k(Enc_2(\mathbf{c}'))$, which implies that the input sequence $\mathbf{c}$ can be uniquely determined given $\boldsymbol{E} \in \mathcal{E}_k(Enc_2(\mathbf{c}))$.*

Proof. We show that $F(\mathbf{c})$ and $F(\mathbf{c}')$ differ in at most $2\lfloor k/d \rfloor$ blocks if $Enc_2(\mathbf{c})$ and $Enc_2(\mathbf{c}')$ can result in the same read matrix $\boldsymbol{E} \in \mathcal{E}_k(Enc_2(\mathbf{c}))$. Then, $F(\mathbf{c})$ and $F(\mathbf{c}')$ cannot have the same $RS_{2\lfloor k/d \rfloor}(S(F(\mathbf{c}))) = RS_{2\lfloor k/d \rfloor}(S(F(\mathbf{c}')))$. Suppose on the contrary, $F(\mathbf{c})$ and $F(\mathbf{c}')$ differ in at least $2\lfloor k/d \rfloor + 1$ blocks. Note that each minimum edit isolated interval has length at most $k(d-1)t \leq B'$. Otherwise it cannot be minimum. Moreover, by Lemma 10 the interval $[b_{1j}, b_{2j}]$ has length at most $B' - k$. Hence, any minimum edit isolated interval, either in $Enc_2(\mathbf{c})$ or $Enc_2(\mathbf{c}')$ in order to obtain $\boldsymbol{E}$, spans at most two blocks of $Enc_2(\mathbf{c})$ or $Enc_2(\mathbf{c}')$, respectively. In addition, the bits of $Enc_2(\mathbf{c})$ or $Enc_2(\mathbf{c}')$ that fall within interval $[b_{1j}, b_{2j}]$ for some j after errors are covered by at most two blocks. Let the at least $2\lfloor k/d \rfloor + 1$ blocks where $F(\mathbf{c})$ and $F(\mathbf{c}')$ differ contain bits that fall within M' intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$. Then, at least $2\lfloor k/d \rfloor + 1 - 2M'$ of these blocks intersect with at least $\lfloor k/d \rfloor + 1 - M'$ minimum edit isolated intervals, either in $Enc_2(\mathbf{c})$ or $Enc_2(\mathbf{c}')$, that are not contained in any $[b_{1j}, b_{2j}]$, $j \in [1, J]$. According to Proposition 4, for any of the M' intervals $[b_{1j}, b_{2j}]$ within which the at least $2\lfloor k/d \rfloor + 1$ different blocks contain bits, the number of errors in the edit isolated intervals associated with interval $[b_{1j}, b_{2j}]$ in $Enc_2(\mathbf{c})_{\mathcal{I}}$ and $Enc_2(\mathbf{c}')_{\mathcal{I}}$ are at least $d - d^*$ and $d + d^*$, or $d + d^*$ and $d - d^*$, respectively, for some $d^* \geq 1$, or both at least d. In either case, the sum of number of errors is at least $2d$. In addition, in any minimum isolated interval that is not contained in $[b_{1j}, b_{2j}]$ after errors, either in $Enc_2(\mathbf{c})$ or $Enc_2(\mathbf{c}')$, the number of errors is at least $2d$. Hence, the total number of errors in $Enc_2(\mathbf{c})$ and $Enc_2(\mathbf{c}')$ should be at least $2dM' + 2d(\lfloor k/d \rfloor + 1 - M') > 2k$, a contradiction to the fact that $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c}, t)$ and $\boldsymbol{E} \in \mathcal{E}_k(\mathbf{c}', t)$. □

Next, we present the decoding algorithm. Similar to what we did in the proof of Theorem 3 and Theorem 4, we use the first row in $\boldsymbol{E}$ to decode $R_2'(\mathbf{c})$. From Lemma 4, we conclude that we can first recover $H(R_2'(\mathbf{c}))$ and then $R_2'(\mathbf{c})$ using the deletion correcting mapping in Lemma 3.

Then, We use Algorithm 4 to obtain a set of output intervals $[b_{1j}, b_{2j}]$ for $j \in [1, J]$. By Lemma 10, the output intervals $[b_{1j}, b_{2j}]$ for $j \in [1, J]$ satisfy properties **(A)**, **(B)**, **(C)**, **(D)**, **(E)**. Then, we use Algorithm 7 to recover the bits $Enc_2(\mathbf{c})_i$ that do not fall within output intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$, after errors, where the indices i can be determined by Algorithm 7. The correctness of Algorithm 7 is guaranteed by Lemma 12. Next, we apply Algorithm 7 to every output interval $[b_{1j}, b_{2j}]$ and $\boldsymbol{E}_{[1,d],[b_{1j},b_{2j}]}$ and obtain an estimate sequence $\mathbf{e}^j$ for $j \in [1, J]$. The sequence $\mathbf{e}^j$ is an estimate of the bits $Enc_2(\mathbf{c})_i$ that fall within intervals $[b_{1j}, b_{2j}]$ after errors, i.e., the bits $Enc_2(\mathbf{c})_i$ that are not addressed in Algorithm 6. According to Lemma 12, the errors in interval $\mathcal{I}_j$, $j \in [1, J]$, where $\mathcal{I}_j$ is the edit isolated interval associated with $[b_{1j}, b_{2j}]$, can be recovered if $|(\boldsymbol{\delta}_1 \cup \boldsymbol{\gamma}_1) \cap \mathcal{I}_j| < d$.

Note that there are at most $\frac{k}{d}$ intervals $[b_{1j}, b_{2j}]$ such that the errors in $\mathcal{I}_j$ are not recovered by Algorithm 7. Moreover, there are at most $J \leq k$ intervals $[b_{1j}, b_{2j}]$, $j \in [1, J]$. We enumerate all possibilities of the set of intervals among $\{[b_{1j}, b_{2j}]\}_{j=1}^{J}$ that are not corrected by Algorithm 7. There are at most $\sum_{i=0}^{\frac{k}{d}} \binom{i}{J} \leq k^{k+1}$ such choices. For each choice $\{[b_{1j_i}, b_{2j_i}]\}_{i=1}^{M}$, we assume that the set of the chosen intervals $\{[b_{1j_i}, b_{2j_i}]\}_{i=1}^{M}$, which cover at most $2M$ blocks in $F(\mathbf{c})$ and thus in $S(F(\mathbf{c}))$, are corrupted by erasures. In addition, we assume that there are at most $\frac{k}{d} - M$ block substitutions in $F(\mathbf{c})$ and thus in $S(F(\mathbf{c}))$. Then we apply the Reed-Solomon decoder [14] to correct at the most $2M$ erasures and at most $\frac{k}{d} - M$ block substitutions in $S(F(\mathbf{c}))$. In addition, we calculate the total number of errors needed to generate the read matrix $\boldsymbol{E}$, when intervals $\{[b_{1j_i}, b_{2j_i}]\}_{i=1}^{M}$ are chosen, by using Proposition 4. We discard the choice of intervals $\{[b_{1j_i}, b_{2j_i}]\}_{i=1}^{M}$, either when Reed-Solomon decoding of $S(F(\mathbf{c}))$ is unsuccessful, or when the total number of errors needed to generate $\boldsymbol{E}$ is greater than k. By Proposition 5, there is a unique and correct choice of the intervals $\{[b_{1j_i}, b_{2j_i}]\}_{i=1}^{M}$ that is not discarded. Therefore, we correctly recover $S(F(\mathbf{c}))$ given $R_2'(\mathbf{c})$, by using the Reed-Solomon decoder [14]. Finally, we use $S(F(\mathbf{c}))$ to recover $F(\mathbf{c})$, and then $\mathbf{c}$ in the same manner as in Section V. The time complexity is dominated by the time needed to compute $S(F(\mathbf{c}))$, which is $O(nt^{2k+1})$.

For cases when $d \leq k \leq 2d - 1$. The codes are similar to those in Section III, which is given by

$$Enc_1(\mathbf{c}) = (F(\mathbf{c}), R_1^{'}(\mathbf{c}), R_1^{''}(\mathbf{c})) \tag{46}$$

where

$$\begin{aligned} R_1^{'}(\mathbf{c}) &= ER(S(F(\mathbf{c}))), \\ R_1^{''}(\mathbf{c}) &= Rep_{k+1}(H(R_1'(\mathbf{c}))). \end{aligned} \tag{47}$$

Similar to cases when $k \geq 2d$, we split $F(\mathbf{c})$ into blocks of length B', the same as in (45). The redundancy is $8k \log t + o(\log t)$. The correctness of the code is similar to cases when $k \geq 2d$. Note that when $d \leq k \leq 2d-1$, there is no minimum edit isolated interval that is not within some interval $[b_{1j}, b_{2j}]$. In addition, there is at most one interval $[b_{1j}, b_{2j}]$ where at least d errors occur. We enumerate all choices of such interval. Similar to the case when $k \geq 2d$, to satisfy the number of errors requirement, only the correct choice of the interval gives a valid and correct decoded sequence $\mathbf{c}$.

VII. Conclusions

We constructed d-head k-deletion racetrack memory codes for any $k \geq d$, extending previous works which addressed cases when $k \leq d$. We proved that for small head distances $t = n^{o(1)}$ and for $k \geq 2d$, the redundancy of our codes is asymptotically at most four times the optimal redundancy. We also generalized the results and proved that the same redundancy results hold for d-head codes correcting a combination of at most k deletions and insertions. Finding a lower bound on the redundancy for $d \leq k \leq 2d - 1$ would be interesting, for both deletion correcting codes and codes correcting a combination of deletions and insertions. It is also desirable to tighten the gap between the upper and lower bounds of the redundancy for cases when $k \geq 2d$.

References

[1] J. Brakensiek, V. Guruswami, and S. Zbarsky, “Efficient low-redundancy codes for correcting multiple deletions,” in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms* (SODA), pp. 1884–1892, 2016

[2] Y. Chee, H. Kiah, A.Vardy, V. Vu and E. Yaakobi, “Codes Correcting limited-shift errors in racetrack memories,” in *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 96–100, 2018.

[3] Y. Chee, H. Kiah, A.Vardy, V. Vu and E. Yaakobi “Coding for racetrack memories,” in *IEEE Transactions on Information Theory*, vol. 64, no. 11, pp. 7094-7112, 2018.

[4] Y. Chee, R. Gabrys, A. Vardy, and V. K. Vu, and E. Yaakobi, “Reconstruction from deletions in racetrack memories,” in *Proc. IEEE Inform. Theory Workshop*, 2018.

[5] K. Cheng, Z. Jin, X. Li, and K. Wu, “Deterministic document exchange protocols, and almost optimal binary codes for edit errors,” *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 200-211, 2018

[6] M. Hayashi, L. Thomas, R. Moriya, C. Rettner and S. S. Parkin, “Current-controlled magnetic domain-wall nanowire shift register,” in *Science*, vol. 320, no. 5873, pp. 209–211, 2008.

[7] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[8] V. I. Levenshtein, “Reconstructing objects from a minimal number of distorted patterns,” (in Russian), in *Dokl. Acad. Nauk* vol. 354 pp. 593-596; English translation, in *Doklady Mathematics*, vol. 55 pp. 417-420, 1997.

[9] S. S. Parkin, M. Hayashi, and L. Thomas, “Magnetic domain-wall racetrack memory,” in *Science*, vol. 320, no. 5873, pp. 190–194, 2008.

[10] J. Sima and J. Bruck, “On optimal k-deletion correcting codes,” *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3360-3375, 2020.

[11] J. Sima, R. Gabrys, and J. Bruck, “Optimal systematic t-deletion correcting codes,” in *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 769–774, 2020.

[12] W. Song, N. Polyanskii, K. Cai, and X. He, ”On Multiple-Deletion Multiple-Substitution Correcting Codes,” in *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 2655–2660, 2021.

[13] Z. Sun, W. Wu and H. Li, “Cross-layer racetrack memory design for ultra high density and low power consumption,” in *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, pp. 1–6, 2013.

[14] T. K. Truong, I. S. Hsu, W. L. Eastman, and I. S. Reed, ”Simplified procedure for correcting both errors and erasures of Reed-Solomon code using Euclidean algorithm, ” in *IEE Proceedings E (Computers and Digital Techniques)*, vol. 135, no. 6, pp.318–324, 1988

[15] A. Vahid, G. Mappouras, D. J. Sorin and R. Calderbank, ”Correcting two deletions and insertions in racetrack memory,” in *arXiv preprint arXiv:1701.06478*, 2017.

[16] R. Venkatesan, V. Kozhikkottu, C. Augustine, A. Raychowdhury, K. Roy and A. Raghunathan “TapeCache: A high density, energy efficient cache based on domain wall memory,” in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pp. 185–190, 2012.

[17] L. R. Welch and E. R. Berlekamp, “Error correction for algebraic block codes,” *US Patent Number 4,633,470*, December 1986.

[18] C. Zhang, G. Sun, X. Zhang, W. Zhang, W. Zhao, T. Wang, Y. Liang, Y. Liu, Y. Wang and J. Shu, “Hi-fi playback: Tolerating position errors in shift operations of racetrack memory,” in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3, pp. 694–706, 2015.

Jin Sima received a B.Eng. and a M.Sc. in electronic engineering from Tsinghua University, China, in 2013 and 2016 respectively, and a Ph.D in electrical engineering from California Institute of Technology (Caltech) in 2022. He is currently a postdoctoral researcher in the Department of electrical and computer engineering at University of Illinois Urbana-Champaign. His research interests include information and coding theory and their applications, with its applications in data storage systems. He is a recipient of the 2019 IEEE Jack Keil Wolf ISIT Student Paper Award, the 2020-2021 IEEE Communication Society Data Storage Best Paper Award, the 2022 Caltech Charles Wilts Prize for best doctoral thesis.

Jehoshua Bruck (Life Fellow, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from the TechnionIsrael Institute of Technology in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from Stanford University in 1989.

He is currently the Gordon and Betty Moore Professor of computation and neural systems and electrical engineering at the California Institute of Technology (Caltech). His current research interests include information theory and systems and the theory of computation in nature. His industrial and entrepreneurial experiences include working with IBM Research, where he has participated in the design and implementation of the first IBM parallel computer, cofounding and serving as the Chairman for Rainfinity (acquired in 2005 by EMC), that created the first virtualization solution for network attached storage; cofounding and serving as the Chairman for XtremIO (acquired in 2012 by EMC), a start-up company that created the first scalable all-flash enterprise storage system; and cofounding and serving as the Chairman for MemVerge, a start-up company that is pioneering big memory computing.

Dr. Bruck was a recipient of the Feynman Prize for Excellence in Teaching, the Sloan Research Fellowship, the National Science Foundation Young Investigator Award, the IBM Outstanding Innovation Award, and the IBM Outstanding Technical Achievement Award.